1 Der Rechner wird vorbereitet

Sie lernen in diesem Kapitel,

- *welche Hilfsmittel Sie für das Schreiben eigener Apps benötigen,*
- *wo Sie diese Hilfsmittel auf der Buch-CD und im Internet finden,*
- *wie Sie das Java-SDK installieren,*
- *wie Sie das Android-SDK installieren,*
- *wie das Android-SDK aufgebaut ist und welche Dokumentationen und Referenzen es beinhaltet,*
- *wie Sie Eclipse installieren und für die Programmierung von Apps vorbereiten.*

Bevor Sie mit der App-Programmierung beginnen können, müssen Sie sicherstellen, dass Sie das nötige Arbeitsgerät zur Verfügung haben. Die gute Nachricht ist: Alles, was Sie zum Schreiben eigener Apps benötigen, gibt es kostenlos im Internet bzw. auf der Buch-CD. Die weniger gute Nachricht ist: Sofern Sie nicht bereits mit Java und Eclipse arbeiten und beides aktuell gehalten haben, steht Ihnen ein kleiner Installationsmarathon bevor. Aber wie sagt schon ein auf Seneca zurückgehendes Sprichwort:

per aspera ad astra

(»Durch Mühsal zu den Sternen« oder wie es in Anlehnung an Hesiod heißt: »Vor den Erfolg haben die Götter den Schweiß gestellt.«)

1.1 Die nötigen Hilfsmittel und Vorbereitungen

Um Apps schreiben zu können, benötigen Sie:

- das **JDK für Java SE**
- das **Android-SDK**
- **Eclipse** mit dem **Android-Plugin** (optional, aber empfehlenswert)
- ein Android-Smartphone (optional)

und natürlich

- einen nicht zu alten Rechner mit mindestens 700 Mbyte freien Festplattenspeicher (optimal wären 2 bis 3 Gbyte) und einem geeigneten Betriebssystem (z.B. Windows XP, Vista, Windows 7 oder Ubuntu Linux[1])
- Spaß am Programmieren, Ausdauer und auch ein bisschen Mut

JDK und Eclipse finden Sie übrigens auf der Buch-CD. Wir hätten gerne noch ein kostenloses Android-Smartphone hinzugepackt, aber es ließ sich leider kein Sponsor finden.

1 64-Bit-Linux-Systeme müssen dafür eingerichtet sein, dass sie auch 32-Bit-Software ausführen können. Unter Ubuntu müssen Sie dazu z.B. vorab das ia32-libs-Paket installieren. Für nähere Erläuterungen konsultieren Sie am besten die Dokumentation zu Ihrem Betriebssystem.

Das Android-Plugin müssen Sie in der aktuellen Version aus dem Internet herunterladen. Die Android-SDK-Version muss auf die Plugin-Version abgestimmt sein, weswegen es sehr zu empfehlen ist, das SDK ebenfalls in der aktuellen Version aus dem Internet herunterzuladen. So gibt es bei der Installation die wenigsten Schwierigkeiten. Nachteilig an diesem Verfahren ist, dass Sie angesichts der kurzlebigen Produktzyklen auf diese Weise mit ziemlicher Sicherheit mit einem anderen Plugin arbeiten als wir beim Schreiben dieses Buches. Lassen Sie sich also nicht zu sehr verwirren, wenn eine Abbildung zum Eclipse-Android-Designer mal nicht hundertprozentig mit Ihrem Bildschirm übereinstimmt oder ein Eclipse-Android-Menübefehl sich geändert hat.

Das JDK für Java SE

Android-Apps werden in der Programmiersprache Java geschrieben. Es sind also, wenn man so möchte, letzten Endes Java-Programme, und, um Java-Programme schreiben zu können, benötigen Sie die Java-Werkzeugkiste für Entwickler, das sogenannte Java-SDK oder JDK.

*SDK ist ein Akronym für **S**oftware **D**evelopment **K**it. Das Kürzel JDK steht für **J**ava **D**evelopment **K**it. Beide Begriffe werden üblicherweise synonym verwendet.*

Das Android-SDK

Der Star unter den Hilfsmitteln, die wir für die Android-Programmierung benötigen, ist natürlich das Android-SDK. Er versorgt uns unter anderem mit Bibliotheken, die wir für die Programmierung benötigen, Hilfsprogrammen zum Erstellen der Apps und einen Emulator, mit dessen Hilfe wir Apps testen können.

Die Eclipse-IDE

Für die Programmierung werden in der Regel viele spezialisierte Hilfsprogramme benötigt, beispielsweise ein Editor zum Aufsetzen der Programmquelltexte, ein Compiler zum Übersetzen der Quelltexte in Programmcode, ein Debugger zur schrittweisen Ausführung von Programmen zwecks Fehleranalyse, um nur die wichtigsten Helfer zu nennen. In integrierten Entwicklungsumgebungen (kurz IDEs) werden diese Hilfsprogramme in einem übergeordneten Programm zusammengefasst. Dies erleichtert die Arbeit, erspart dem Programmierer ständige Wechsel zwischen den Anwendungen und erlaubt zudem eine intensivere Zusammenarbeit der eingesetzten Hilfsprogramme (wie z.B. die Zuordnung von Compiler-Fehlermeldungen zu

Verwechseln Sie das JDK nicht mit der JRE. JRE ist das Akronym für Java Runtime Environment (Java-Laufzeitumgebung). Java-Programme können nur auf Rechnern ausgeführt werden, auf denen eine JRE installiert ist. Die JRE dient aber wirklich nur der Ausführung von Java-Programmen. Für das Schreiben und Entwickeln von Java-Programmen reicht die JRE nicht aus, da ihr wichtige Hilfsprogramme wie z.B. der Java-Compiler javac fehlen. (Im JDK ist eine JRE mit enthalten.)

Quelltextzeilen oder die Anzeige der aktuellen Ausführungsposition beim Debuggen im Editor).

Eclipse ist eine solche integrierte Entwicklungsumgebung und unter professionellen Programmierern recht beliebt, denn Eclipse ist kostenfrei (für viele Unternehmen und Programmierer das wichtigste Argument überhaupt), recht leistungsstark und vor allem sehr flexibel. Eclipse kann nämlich durch Installation entsprechender Plugins für die Erstellung praktisch beliebiger Anwendungen in beliebigen Sprachen – und dank des Android-Plugin eben auch für die Erstellung von Android-Apps – hergerichtet werden.

Die Arbeit mit Eclipse und dem Android-Plugin ist leider nicht immer so intuitiv und einfach, wie man sich dies, vor allem als Programmierneuling oder Umsteiger von anderen Entwicklungsumgebungen, wünschen würde. Wir werden daher gerade in den ersten Kapiteln nebenbei auch des Öfteren auf die Arbeit mit Eclipse eingehen. Der Aufwand lohnt allemal, denn die Alternative – Android-Programmierung mit dem reinen Android-SDK – birgt noch weitaus mehr Tücken.

Die wichtigsten Arbeitsschritte mit Eclipse haben wir für Sie noch einmal im Anhang zusammengefasst!

1.2 Das JDK für Java SE

Stufe 1: Java (1. Phase)

Wir werden Ihnen nun zeigen, wie Sie Ihren Rechner in vier Stufen für die Android-Programmierung fit machen können. Stufe 1 ist die Installation des JDK.

Halten Sie sich an die von uns vorgegebene Installationsreihenfolge. Sonst gibt es möglicherweise Ärger, wenn die Setupprogramme von Android und Eclipse nach einer JDK-Installation auf Ihrem Rechner suchen und keine vorfinden.

Falls Sie aber das JDK in der Version 6 oder 7 bereits auf Ihrem Rechner installiert haben (siehe auch den Abschnitt »Test«), können Sie die nachfolgenden Erläuterungen natürlich überspringen und gleich zu Stufe 2, der Installation des Android-SDK, übergehen.

Linux- und Windows-Anwender finden auf der Buch-CD passende Setupdateien für das JDK 6u25 (Version 6, Update 25) sowohl für 32-Bit- als auch 64-Bit-Systeme. Die jeweils aktuelle Version des JDK können Sie übrigens von der Oracle-Website herunterladen: http://www.oracle.com/technetwork/java/javase/downloads/index.html.

Sun, Oracle und das OpenJDK

Vielleicht haben Sie im Internet oder in der einschlägigen Presse etwas darüber gelesen, dass die Sprache Java von Sun an Oracle verkauft wurde, und sind verunsichert, da es nun möglicherweise mehrere – konkurrierende? – JDK für die Programmierung mit Java gibt. Lassen Sie uns hierzu ein paar Worte sagen.

Die Programmiersprache Java ist im Grunde genommen nichts weiter als eine technische Spezifikation, die beschreibt, wie Java-Code aussehen soll. Lebendig wird diese Spezifikation erst durch eine Entwicklungsumgebung, das JDK, das dem Programmierer die Mittel an die Hand

gibt, um in Java geschriebene Quelltexte in lauffähige Java-Programme umzuwandeln.

Früher oblagen Pflege und Weiterentwicklung der Sprache Java und des zugehörigen JDK ganz der Firma Sun Microsystems, in deren Auftrag die Sprache Java überhaupt erst entwickelt wurde. Doch Sun hat sich mittlerweile aus diesem Aufgabenbereich weitestgehend zurückgezogen. Bereits 2006 begann man mit der Umwandlung des JDK in ein Open-Source-Projekt, wenige Jahre später wurden dann die weiteren Rechte an Java an die Firma Oracle verkauft.

Seitdem gibt es zwei wichtige Referenzimplementierungen: das OpenJDK und das Oracle JDK. Oracle hat zugesichert, das OpenJDK weiter zu unterstützen, sodass derzeit nicht zu befürchten ist, dass die JDK-Versionen stark auseinanderdriften werden. Zudem wird das JDK für Java SE derzeit von Oracle weiter kostenfrei zum Download angeboten und darf im Rahmen der Lizenzbedingungen genutzt werden. Lediglich für diverse kommerzielle Zusatzkomponenten, die optional hinzugenommen werden können, sind Lizenzgebühren zu bezahlen.

1.2.1 Installation

Die reine Installation des Java Development Kit (JDK) ist schnell erledigt und relativ problemlos. Etwas schwieriger – zumindest für Anwender, die dies noch nie gemacht haben – ist die Eintragung der Java-Werkzeuge in den Systempfad. Doch eins nach dem anderen.

JDK schon vorhanden?

Sie sind unsicher, ob auf Ihrem System nicht vielleicht schon ein passendes JDK installiert ist? Dann rufen Sie ein Konsolenfenster auf (siehe hierzu auch den Abschnitt »Test« weiter unten) und schicken Sie den folgenden Befehl ab:

```
javac -version
```

Ist ein Oracle- oder Sun-JDK installiert, wird daraufhin die Versionsnummer des Java-Compilers angezeigt. Beginnt diese mit 1.6 oder 1.7, können Sie die vorhandene JDK-Installation verwenden und zur Installation des Android-SDK übergehen.

Lautet die Ausgabe

```
javac: no input files
```

bedeutet dies vermutlich, dass das OpenJDK installiert ist. In diesem Fall müssen Sie die Versionsnummer des Java-Interpreters abfragen, und zwar mit dem Befehl `java -version`.

1. Führen Sie die Setupdatei aus.

   Windows: Wenn Sie mit einem 32-Bit-Rechner arbeiten und wie empfohlen die Installationsdateien von der Buch-CD benutzen möchten, starten Sie die Setupdatei *jdk-6u25-windows-i586.exe*, besitzen Sie dagegen einen 64-Bit-Rechner, starten Sie die Setupdatei *jdk-6u25-windows-x64.exe* – z.B. durch Doppelklick auf die Datei im Windows Explorer, Aufruf mit ⊞+E.

   Linux: Legen Sie ein Verzeichnis an, unter dem das JDK installiert werden soll (beispielsweise *Java*), und kopieren Sie in dieses die Datei *jdk-6u25-linux-i586.bin* für 32-Bit-Systeme und *jdk-6u25-linux-x64.bin* für 64-Bit-Systeme. Danach öffnen Sie ein Konsolenfenster, wechseln in das angelegte Verzeichnis und lassen die Datei ausführen, z.B. mit:

   ```
   ./jdk-6u25-linux-i586.bin
   ```

   oder mit zuvor gesetzten Rechten:

   ```
   chmod +x jdk-6u25-linux-i586.bin
   ./jdk-6u25-linux-i586.bin
   ```

2. Folgen Sie den Anweisungen des Setupprogramms und lassen Sie das JDK möglichst vollständig installieren.

   Windows: Das Setupprogramm führt Sie durch mehrere Dialoge. Im Dialog Custom Setup können Sie auswählen, welche Komponenten Sie installieren möchten (siehe Abbildung 1.1). Über die Schaltfläche Change können Sie zudem ein anderes Installationsverzeichnis auswählen. Wir empfehlen allerdings, das JDK komplett zu installieren und auch das vorgegebene Installationsverzeichnis beizubehalten.

   Das Installationsverzeichnis für die JRE, sofern Sie diese mitinstallieren lassen, wird gegen Ende der Installation noch einmal extra abgefragt.

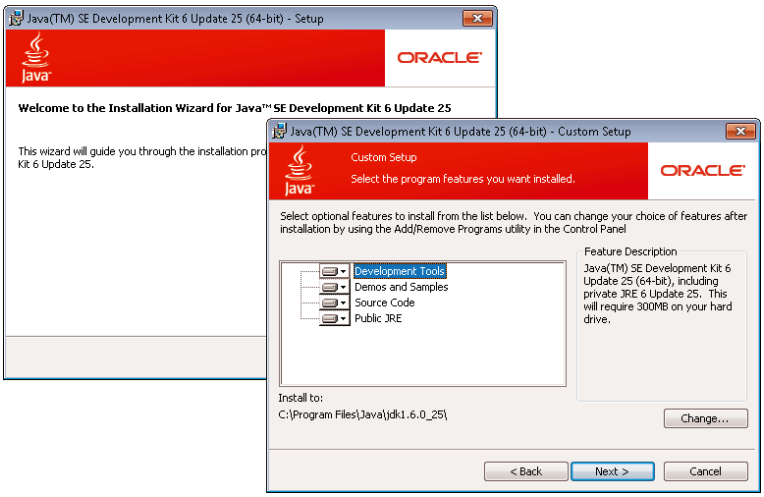

Java(TM) SE Development Kit 6 Update 25 (64-bit) - Setup
ORACLE
Welcome to the Installation Wizard for Java™ SE Development Kit 6 Update 25
This wizard will guide you through the installation pro
Kit 6 Update 25.
Java(TM) SE Development Kit 6 Update 25 (64-bit) - Custom Setup
Custom Setup
Select the program features you want installed.
ORACLE
Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel
Development Tools
Demos and Samples
Source Code
Public JRE
Feature Description
Java(TM) SE Development Kit 6 Update 25 (64-bit), including private JRE 6 Update 25. This will require 300MB on your hard drive.
Install to:
C:\Program Files\Java\jdk1.6.0_25\
Change...
< Back
Next >
Cancel

Abbildung 1.1:
Windows-Anwender können hier auswählen, welche Komponenten installiert werden sollen und in welchem Verzeichnis das JDK installiert werden soll.

Wenn Sie mit Windows arbeiten und mit Festplattenspeicher knapp sind, deaktivieren Sie die Option Public JRE. Das SDK (Option Development Tools) enthält eine eigene private Version der JRE, die zum Testen der Programme verwendet werden kann. Die Optionen Source Code und Demos können Sie bei Speichermangel ebenfalls deaktivieren. Allerdings spart dies relativ wenig Mbytes, und insbesondere die Quelldateien der Java-API sind für fortgeschrittene Java-Programmierer eine wertvolle Referenzquelle!

Linux: Unter Linux läuft die Installation vollautomatisch ohne Abfragen ab.

3. Zum Abschluss führt das Setupprogramm zu einer Website, wo Sie sich als Benutzer registrieren lassen können (aber nicht müssen).

Deinstallation

Wenn Sie das JDK irgendwann wieder deinstallieren möchten, rufen Sie die mitgelieferte Deinstallationsroutine über die Systemsteuerung, Seite Programme/Programme und Funktionen auf.

Linux-Anwender löschen einfach das Verzeichnis der JDK-Installation.

1.2.2 Eintragung in den Systempfad

Nach der Installation müssen Sie Ihr System noch so konfigurieren, dass die JDK-Hilfsprogramme von überall bequem aufgerufen werden können.

Der Schlüssel hierzu ist die System-Umgebungsvariable PATH. Der Inhalt dieser Variablen ist nichts weiter als eine Sammlung von Verzeichnisangaben, die vom Betriebssystem und vielen anderen Programmen bei Bedarf nach ausführbaren Programmen und Programmbibliotheken durchsucht werden. Indem Sie das BIN-Verzeichnis der Java-Installation in die PATH-Variable eintragen, erleichtern Sie sich nicht nur die direkte Arbeit mit den diversen Java-Werkzeugen (wie z.B. dem Java-Compiler *javac*, siehe *Kapitel 2* des Java-Tutoriums auf der Buch-CD), sondern stellen auch sicher, dass das Android-SDK, das wir in Phase 2 installieren werden, auf diese Werkzeuge zugreifen kann.

Windows

Die Umgebungsvariable PATH wird über den Dialog der Systemeigenschaften verwaltet.

1. Rufen Sie über Start oder Start/Einstellungen die Systemsteuerung auf, schalten Sie diese gegebenenfalls in die klassische Ansicht (Anzeige: Kleine Symbole unter Windows 7) und Sie gelangen via System, Erweiterte Systemeinstellungen (Register Erweitert unter Windows XP), Schaltfläche Umgebungsvariablen zum Ziel.
2. Danach können Sie die Systemvariable im unteren Listenfeld auswählen, zum Bearbeiten laden und den Pfad zu den Java-Programmen anhängen.
3. Springen Sie im Eingabefeld Wert der Variablen an das Ende des aktuellen PATH-Werts und fügen Sie das BIN-Verzeichnis der JDK-Installation hinzu. Zum Beispiel:

   Ende des alten Eintrags:

   ```
   ... C:\xampp\php;
   ```

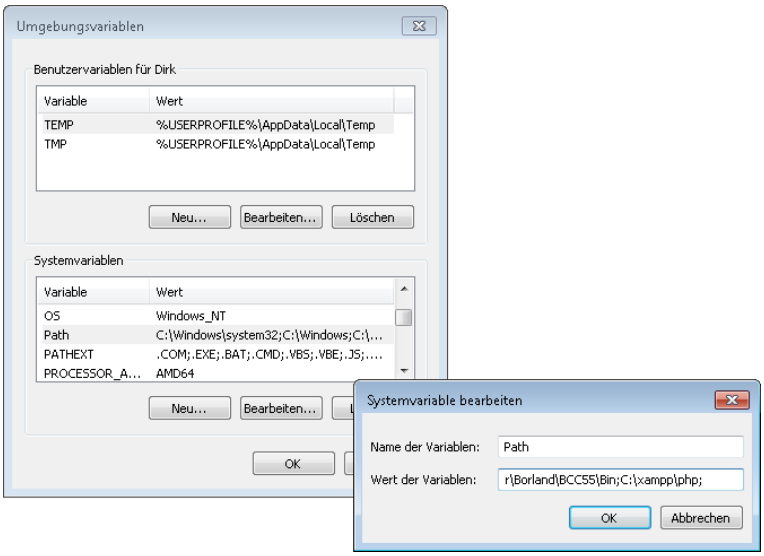


Abbildung 1.2:
Anpassung der PATH-Variablen unter Windows 7 (zeigt das Ende des PATH-Werts, bevor der Pfad zum Java-BIN-Verzeichnis angehängt wird)

Ende des überarbeiteten Eintrags:

```
...C:\xampp\php;C:\Program Files\Java\jdk1.6.0_25\bin;
```

Das Semikolon `;` dient zur Trennung der einzelnen Verzeichnisangaben in der `PATH`-Variablen.

4. Schließen Sie alle Dialoge durch Klick auf OK.

Linux

Suchen Sie die Pfadangabe `path` in der zuständigen INI-Datei (je nach Konfiguration *.login*, *.profile*, *.tcshrc*, *.bashrc* o.ä.) und fügen Sie das Java-BIN-Verzeichnis, das z.B. */home/ihrname/Java/jdk1.6.0_25/bin* lauten könnte, in der nächsten Zeile nach der bisherigen Pfadangabe hinzu.

Für die C-Shell sieht dies beispielsweise wie folgt aus:

```
set path = (/home/ihrname/Java/jdk1.6.25/bin . $path)
```[1]

Für die Bourne-Again-Shell (bash) könnte der Eintrag so lauten:

```
export PATH=/home/ihrname/Java/jdk1.6.25/bin:.:${PATH}
```

oder

```
PATH="/home/ihrname/Java/jdk1.6.25/bin:.:$PATH"
```[2]

1 Mehrere Verzeichnisangaben werden durch Leerzeichen getrennt.
2 Mehrere Verzeichnisangaben werden durch einen Doppelpunkt getrennt.

Um Tippfehler im Verzeichnispfad zu vermeiden, können Sie so vorgehen, dass Sie das BIN-Verzeichnis der Java-Installation zuerst im Windows Explorer[1] *öffnen. Klicken Sie dann oben im Windows Explorer in die Adressleiste, wo jetzt der gewünschte Pfad inklusive bin angezeigt und ausgewählt sein sollte. Drücken Sie* Strg+C*, um den Pfad in die Zwischenablage zu kopieren, wechseln Sie an das Ende des Eingabefelds* Wert der Variablen *und fügen Sie den kopierten Verzeichnispfad mit* Strg+V *ein. Kontrollieren Sie noch einmal den Eintrag und achten Sie vor allem darauf, dass vor und nach dem Eintrag je ein Semikolon steht.*

1 Aufruf über ⊞+E

Wenn Sie gar keine PATH-Angabe finden, klicken Sie auf Neu *und richten Sie eine neue PATH-Variable ein.*

1.2.3 Test

Hat alles geklappt? Machen wir einen kleinen Test, der uns nebenbei auch gleich bestätigt, dass die richtige JDK-Version installiert ist.

Die Konsole

Mehr Informationen zur Bedienung der Windows-Konsole finden Sie als Tutorium zum Herunterladen auf der Website www.carpelibrum.de.)

Für den Test verwenden wir die Konsole. Die Konsole, unter Windows meist »Eingabeaufforderung« genannt, ist eine Schnittstelle zum Betriebssystem, über die Sie Zeile für Zeile Befehle abschicken können. Die Eingabe erfolgt in der jeweils untersten Zeile nach dem sogenannten Prompt (der per Voreinstellung meist das aktuelle Verzeichnis angibt). Jeder Befehl muss durch Drücken der ↵-Taste abgeschickt werden.

Um unter Windows Vista/7 ein Konsolenfenster aufzurufen, klicken Sie auf das Start-Menü und wählen Sie unter Alle Programme\Zubehör den Eintrag Eingabeaufforderung. Unter älteren Windows-Betriebssystemen heißt die Konsole manchmal auch MS-DOS-Eingabeaufforderung oder ist direkt unter Start/Programme zu finden.

Unter Linux heißt die Konsole oft auch Terminal und kann über entsprechende Desktop-Symbole oder Links im Startmenü geöffnet werden.

JDK korrekt installiert?

1. Rufen Sie ein Konsolenfenster auf (siehe oben).
2. Schicken Sie den Befehl `javac -version` ab.

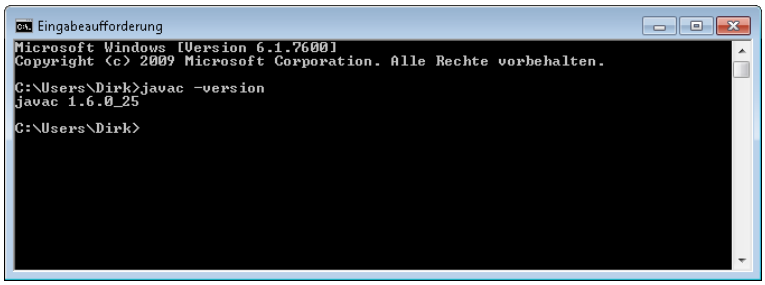


***Abbildung 1.3:** Prima! Die JDK-Installation war erfolgreich.*

Wenn das JDK korrekt installiert und eingerichtet ist, erhalten Sie als Ergebnis eine kurze Meldung, die die Versionsnummer des Java-Compilers *javac* angibt. Beginnt die ausgegebene Versionsnummer mit 1.6 oder 1.7, so sind Sie im grünen Bereich.

Erhalten Sie stattdessen eine Fehlermeldung der Art »Der Befehl javac konnte nicht gefunden werden«, oder können Sie gar nicht erst in das gewünschte Verzeichnis wechseln, ist etwas schiefgegangen.

Höchstwahrscheinlich ist der Eintrag im Systempfad falsch. Sie können dies leicht kontrollieren, denn wenn das JDK korrekt installiert wurde und nur der PATH-Eintrag fehlt oder falsch ist, kann der Java-Compiler *javac* nur direkt aus seinem Verzeichnis heraus ausgerufen werden.

1. Wechseln Sie in das BIN-Verzeichnis Ihrer JDK-Installation.

   Zum Wechseln in ein anderes Verzeichnis verwenden Sie den *cd*-Befehl. Wenn Sie das JDK in das Verzeichnis *C:\Program Files\Java\jdk1.6.0_25* installiert haben, lautet der Befehl:

   ```
   Prompt> cd C:\Program Files\Java\jdk1.6.0_25\bin
   ```

   Unter Linux werden Sie natürlich einen anderen Installationspfad verwendet haben. Vielleicht haben Sie das JDK unter Ihrem Home-Verzeichnis installiert:

   ```
   Prompt> cd /dirk/jdk1.6.0_25/bin
   ```

2. Schicken Sie den Befehl `javac -version` ab.

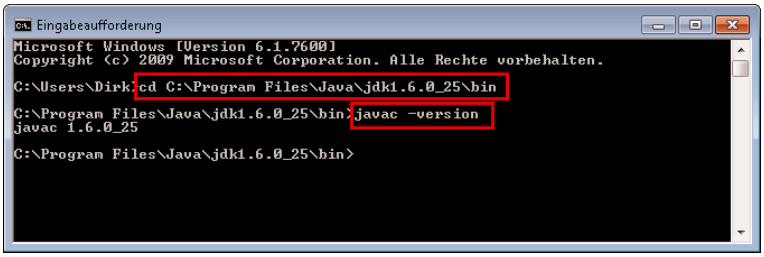


Abbildung 1.4:
Okay, auf jeden Fall ist das JDK korrekt installiert.

Wenn sich jetzt der Compiler meldet, ist das JDK auf jeden Fall korrekt installiert, und der Grund, warum Sie den Compiler nicht von jedem beliebigen Verzeichnis aus aufrufen können, ist im PATH-Eintrag zu suchen.

Kontrollieren Sie noch einmal die Eintragung. Fahren Sie zur Sicherheit den Rechner nach der Bearbeitung des Eintrags neu hoch (sollte unter Windows Vista oder 7 nicht nötig sein, aber man weiß ja nie) und testen Sie, ob Sie *javac* erfolgreich aus anderen Verzeichnissen als dem BIN-Verzeichnis aufrufen können.

1.2.4 Firewall

Wenn Sie eine Firewall installiert haben, wird im Zuge der nächsten Installationsschritte vermutlich irgendwann ein Dialog bezüglich des Java-Hilfsprogramms *javaw* aufspringen. Sorgen Sie dann dafür, dass das Hilfsprogramm nicht weiter blockiert wird.

1.3 Das Android-SDK

Stufe 2: Android

Wir kommen zu Stufe 2: der Installation des Android-SDK.

Das Android-SDK finden Sie nicht auf der Buch-CD, Sie müssen es von der Android-Website herunterladen: *http://developer.android.com/sdk/index.html*.

Der Grund, warum wir das Android-SDK nicht mit auf die Buch-CD gepackt haben, ist, dass die SDK-Version zu der Version des Android-Plugin für Eclipse passen muss, das Sie in Schritt 4 installieren werden. Das Android-Plugin kann aber nur von der Android-Website heruntergeladen werden und wird dort nur in der jeweils aktuellen Version zur Verfügung gestellt. Mit anderen Worten: Wenn Sie beide Pakete, SDK und Plugin, zur etwa gleichen Zeit aus dem Internet herunterladen, gibt es die wenigsten Schwierigkeiten bei der Installation.

1.3.1 Installation

Die Installation des Android-SDK ist schnell erledigt.

1. Laden Sie die Setupdatei von der Android-Website *http://developer.android.com/sdk/index.html* herunter und führen Sie sie aus.

   Windows: Für Windows hieß die Setupdatei bisher üblicherweise *installer_rNN-windows.exe*, wobei »rNN« für die aktuelle Versionsnummer steht. Zum Installieren doppelklicken Sie auf die heruntergeladene Setupdatei.

   Linux: Für Linux hieß die Setupdatei bisher üblicherweise *android-sdk_rNN-linux.tgz*, wobei »rNN« für die aktuelle Versionsnummer steht. Kopieren Sie die Datei in das übergeordnete Verzeichnis, unter dem das Android-SDK installiert werden soll, und entpacken Sie die Datei mit dem Befehl:

   ```
   tar -xzf android-sdk_rNN-linux.tgz
   ```

   Danach können Sie die ZIP-Datei löschen.

2. Folgen Sie den Anweisungen des Setupprogramms und lassen Sie das SDK installieren.

Während wir beim Schreiben des Buches allerdings noch die Version 11 benutzten, berücksichtigt der vorliegende Text die Version 16, also

- installer_r16-windows.exe bzw.
- android-sdk_r16-linux.tgz

Die für die Installation bedeutendste Änderung ist wohl, dass der SDK Manager und der AVD Manager nicht mehr wie im Buch beschrieben in einem Tool-Programm vereinigt sind, sondern nun eigenständige Programme sind.

Windows: Eigentlich müssen Sie sich nur mittels der Next-Schaltfläche durch die Dialoge klicken.

In früheren Versionen gab es das Problem, dass das Setupprogramm im ersten Anlauf das Java-JDK nicht finden konnte. Mittlerweile taucht dieser Fehler unserer Erfahrung nach nicht mehr auf. Falls doch: Drücken Sie einfach die Schaltfläche Back, um einen Dialog zurückzugehen, und anschließend wieder Next. Danach sollte der Dialog aus *Abbildung 1.1* zu sehen sein, der bestätigt, dass das JDK gefunden wurde.

Im weiteren Verlauf können Sie das Installationsverzeichnis auswählen. Kontrollieren Sie dabei den vorgeschlagenen Installationsort. Dieser enthält unter Umständen Leerzeichen im Pfad und auf manchen

Windows-Architekturen führt dies dazu, dass das SDK später nicht vom Eclipse-Android-Plugin verwendet werden kann. Wählen Sie ein Verzeichnis, in dessen Pfad keine Leer- oder Sonderzeichen vorkommen, wie z.B. *C:\Compiler\Android\android-sdk*.

Deaktivieren Sie auf der letzten Seite die Option zum Starten des SDK-Managers, bevor Sie auf FINISH klicken.

Linux: Unter Linux entfällt dieser Schritt.

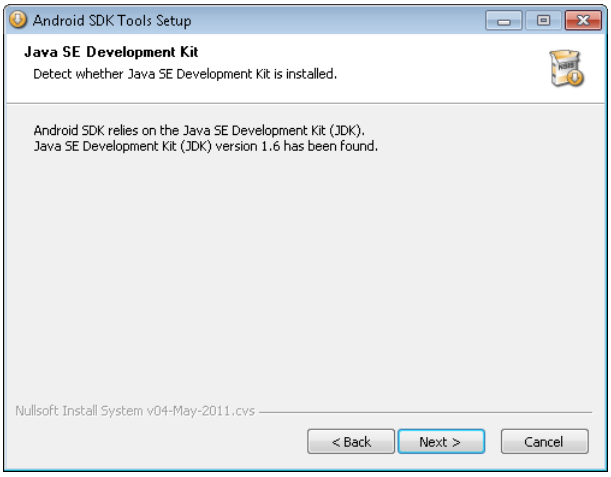

Android SDK Tools Setup
Java SE Development Kit
Detect whether Java SE Development Kit is installed.
Android SDK relies on the Java SE Development Kit (JDK).
Java SE Development Kit (JDK) version 1.6 has been found.
Nullsoft Install System v04-May-2011.cvs
< Back
Next >
Cancel

Abbildung 1.5:
Das Setupprogramm hat eine installierte JDK-Version gefunden.

Der erste Teil des Android-SDK ist nun eingerichtet, doch es fehlen noch diverse Komponenten: wie z.B. die Bibliotheken, die Sie dringend für die Programmierarbeit benötigen. Mithilfe des SDK-Managers können Sie diese nun installieren.

3. Starten Sie den SDK-Manager.

   Windows: Öffnen Sie dazu z.B. den Windows Explorer, wechseln Sie in das Verzeichnis des Android-SDK und starten Sie den SDK-Manager durch Doppelklick auf die Datei *SDK Manager.exe*. (Wenn Sie nicht als Administrator angemeldet sind, klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie im Kontextmenü den Befehl ALS ADMINISTRATOR AUSFÜHREN aus.)

   Linux: Wechseln Sie in das *tools*-Verzeichnis des Android-SDK und starten Sie das Programm *android*.

In dem Fenster des SDK-Managers sehen Sie nun eine größere Auswahl an Komponenten (Packages), die Sie installieren können. Die Komponenten mit dem Häkchen vor dem Namen sind für die Installation vorgemerkt.

! Der SDK-Manager benötigt eine bestehende Internetverbindung!

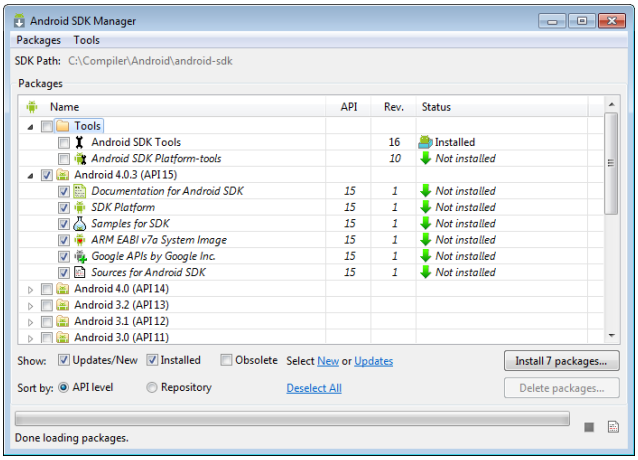

Android SDK Manager
Packages Tools
SDK Path: C:\Compiler\Android\android-sdk
Packages
Name API Rev. Status
Tools
Android SDK Tools 16 Installed
Android SDK Platform-tools 10 Not installed
Android 4.0.3 (API 15)
Documentation for Android SDK 15 1 Not installed
SDK Platform 15 1 Not installed
Samples for SDK 15 1 Not installed
ARM EABI v7a System Image 15 1 Not installed
Google APIs by Google Inc. 15 1 Not installed
Sources for Android SDK 15 1 Not installed
Android 4.0 (API 14)
Android 3.2 (API 13)
Android 3.1 (API 12)
Android 3.0 (API 11)
Show: Updates/New Installed Obsolete Select New or Updates
Install 7 packages...
Sort by: API level Repository
Deselect All
Delete packages...
Done loading packages.

Abbildung 1.6: Der SDK-Manager hat für Sie eine Vorauswahl der Komponenten getroffen, die noch installiert werden sollten.

4. Markieren Sie zusätzlich das Kästchen für die Plattformen Android 2.2, mit der wir in diesem Buch standardmäßig arbeiten.

Plattformen und APIs

Wie von jeder Software kommen auch von dem Android-Betriebssystem ständig neue und erweiterte Versionen (Plattformen) heraus. Zu jeder dieser Plattformen gibt es eine eigene API[1]. Der SDK-Manager lädt per Voreinstellung die Bibliotheken zu allen Android-Versionen herunter, sodass Sie später, wenn Sie beginnen, eine App zu schreiben, frei auswählen können, zu welcher Android-Version die App kompatibel sein soll. Dazu an gegebener Stelle mehr.

1 API ist das Akronym für »Application Programming Interface« und die Schnittstelle zu den Bibliotheksklassen, die Sie für die App-Programmierung benutzen.

! Die Dateien für die einzelnen Plattformen nehmen zusammen recht viel Festplattenspeicher in Anspruch. Wenn Sie mit Festplattenspeicher knapp sind oder die Download-Zeit möglichst kurz halten möchten, entscheiden Sie sich für ein oder zwei Plattformen, beispielsweise 2.2 und 4.0.

5. Klicken Sie auf die INSTALL...-Schaltfläche.
6. Im nachgeschalteten Dialog CHOOSE PACKAGES TO INSTALL aktivieren Sie die Option ACCEPT ALL, um für alle zu installierenden Komponenten die Lizenzvereinbarung anzunehmen (soweit es solche gibt) und klicken Sie dann auf INSTALL, um die ausgewählten Komponenten installieren zu lassen.

Das Herunterladen und Installieren der Komponenten wird vermutlich etwas länger dauern. Haben Sie also ein wenig Geduld. Im automatisch aufspringenden Log-Fenster bzw. anhand des Fortschrittsbalkens im SDK Manager-

Fenster können Sie das Fortschreiten der Installation verfolgen. Geht alles gut, werden die installierten Komponenten angezeigt.

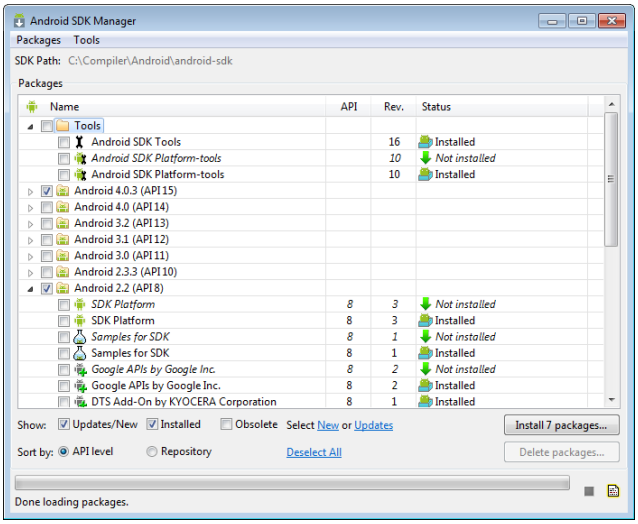

Android SDK Manager
Packages Tools
SDK Path: C:\Compiler\Android\android-sdk
Packages
Name API Rev. Status
Tools
Android SDK Tools 16 Installed
Android SDK Platform-tools 10 Not installed
Android SDK Platform-tools 10 Installed
Android 4.0.3 (API 15)
Android 4.0 (API 14)
Android 3.2 (API 13)
Android 3.1 (API 12)
Android 3.0 (API 11)
Android 2.3.3 (API 10)
Android 2.2 (API 8)
SDK Platform 8 3 Not installed
SDK Platform 8 3 Installed
Samples for SDK 8 1 Not installed
Samples for SDK 8 1 Installed
Google APIs by Google Inc. 8 2 Not installed
Google APIs by Google Inc. 8 2 Installed
DTS Add-On by KYOCERA Corporation 8 1 Installed
Show: Updates/New Installed Obsolete Select New or Updates
Install 7 packages...
Sort by: API level Repository
Deselect All
Delete packages...
Done loading packages.

Abbildung 1.7:
Die für die Installation vorgesehenden Pakete (Status Not installed) wurden installiert (gedoppelter Eintrag mit Status Installed). Nach einen Neustart des SDK Managers werden die veralteteten Not Installed-Einträge nicht mehr angezeigt.

7. Schließen Sie die Fenster des SDK-Managers.

Deinstallation

Wenn Sie das Android-SDK irgendwann wieder deinstallieren möchten, wechseln Sie in das Installationsverzeichnis (standardmäßig *C:\Compiler\Android/android-sdk*) und rufen Sie dort das Programm *uninstall.exe* auf. Anschließend können Sie von Hand noch das Installationsverzeichnis selbst löschen.

Linux-Anwender löschen einfach das Verzeichnis der Android-Installation.

1.3.2 Dokumentation und API-Referenz

Anstelle eines Tests lassen Sie uns kurz durch das Installationsverzeichnis des Android-SDK streifen, um zu sehen, ob alle wichtigen Komponenten heruntergeladen wurden.

Hier finden Sie Hilfe

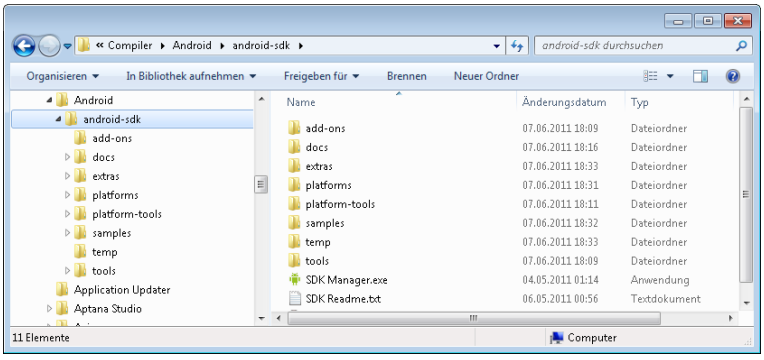


Abbildung 1.8:
Das Verzeichnis des Android-SDK

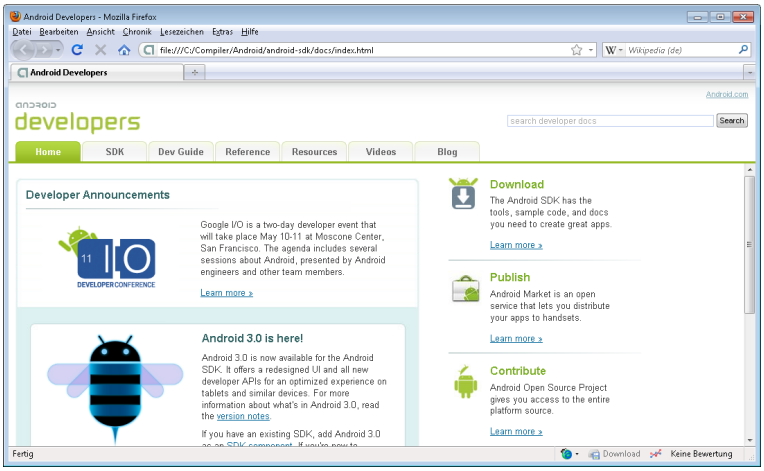
Tabelle 1.1: Wichtige Verzeichnisse des Android-SDK

Unterverzeichnis	Inhalt
tools	Hier sind die verschiedenen Hilfsprogramme des SDK zusammengefasst; darunter z.B. auch der Emulator, der ein Android-Smartphone simuliert und mit dessen Hilfe Sie Ihre Apps direkt auf dem PC testen können.
platforms	Für jede Plattform (Android-Version), die Sie mithilfe des SDK-Managers heruntergeladen haben, finden Sie hier ein eigenes Unterverzeichnis mit der Programmierbibliothek (*android.jar*), individuellen Oberflächen für den Emulator (*skins*) und diversen anderen plattformspezifischen Dateien.
platform-tools	Enthält plattformspezifische Hilfsprogramme.
samples	Diverse Beispielprogramme, aufgeteilt nach Plattformen (Android-Versionen)
docs	Hilfe und Dokumentation

Die Android-Hilfe ist komplett auf Englisch.

Besondere Aufmerksamkeit verdient das Verzeichnis *docs*. Hier finden Sie die vollständige Dokumentation zu Android, inklusive Programmierhandbüchern und API-Referenz, quasi ein lokales Abbild der Android-Website. Ausgangspunkt ist die Datei *index.html*.

Abbildung 1.9: Die Startseite index.html der Android-Hilfe


Android Developers - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
file:///C:/Compiler/Android/android-sdk/docs/index.html
Android Developers
developers
Home
SDK
Dev Guide
Reference
Resources
Videos
Blog
Developer Announcements
Google I/O is a two-day developer event that will take place May 10-11 at Moscone Center, San Francisco. The agenda includes several sessions about Android, presented by Android engineers and other team members.
Learn more »
Android 3.0 is here!
Android 3.0 is now available for the Android SDK. It offers a redesigned UI and all new developer APIs for an optimized experience on tablets and similar devices. For more information about what's in Android 3.0, read the version notes.
Download
The Android SDK has the tools, sample code, and docs you need to create great apps.
Learn more »
Publish
Android Market is an open service that lets you distribute your apps to handsets.
Learn more »
Contribute
Android Open Source Project gives you access to the entire platform source.
Learn more »
Fertig
Keine Bewertung

Seite	Inhalt
SDK	Hier können Sie sich über die Besonderheiten der diversen Android-Plattformen informieren. Über den Link *SDK System Requirements*, rechts unten im Abschnitt *More Information*, können Sie die Systemvoraussetzungen einsehen, die für eine erfolgreiche Installation des Android-SDK Bedingung sind.
Dev Guide	Hier finden Sie Hinweise, Artikel und Tutorien zu praktisch allen Aspekten der Android-Programmierung.
Reference	Dies ist die Referenz der Elemente aus der Android-Bibliothek. Die Referenz ist analog der Java-API-Dokumentation aufgebaut. Zuerst wählen Sie links oben ein Paket (*Package*) aus, dann links unten eine der im Paket enthaltenen Klassen (*Classes*) oder sonstigen Elemente. Danach wird im rechten Bereich die Dokumentation zu dem ausgewählten Element angezeigt.

Tabelle 1.2:
Wichtige Seiten der Android-Hilfe

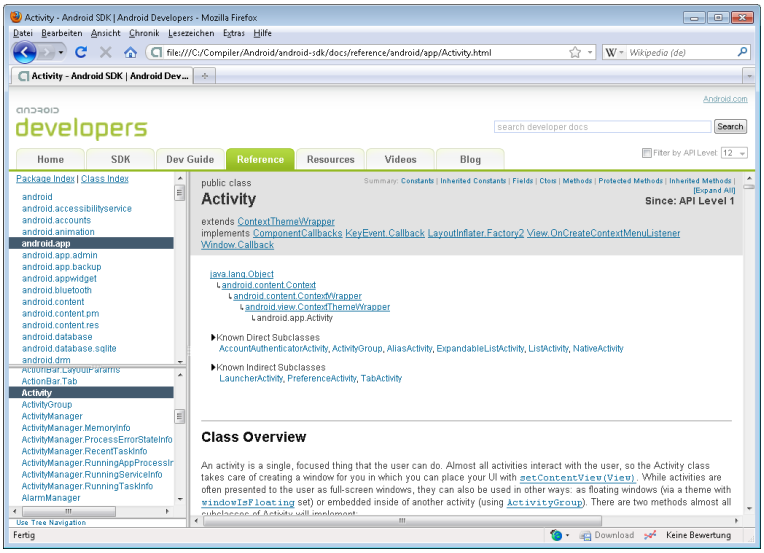


Abbildung 1.10:
Referenzdokumentation zur Android-Klasse Activityw

1.4 Eclipse

Stufe 3: Eclipse (1. Phase)

Stufe 3: Mit Eclipse steht Ihnen eine komplette und professionelle Entwicklungsumgebung zur Verfügung. Die Arbeit mit Eclipse erfordert zwar ein wenig Übung und Eingewöhnung, aber dafür bietet sie auch eine großartige Unterstützung und erleichtert vieles.

Linux- und Windows-Anwender finden auf der Buch-CD passende Setupdateien für Eclipse (Version Helios, geeignet für Android-SDK 11). Die jeweils aktuelle Version können Sie von der Eclipse-Website herunterladen: http://www.eclipse.org/downloads.

1.4.1 Installation

Sie fanden, dass die Installation des JDK und des Android-SDK eigentlich recht einfach war? Dann werden Sie vielleicht erstaunt sein, dass die Installation von Eclipse noch einfacher ist – Sie müssen lediglich die komprimierte Eclipse-Datei von der Buch-CD extrahieren.

1. Extrahieren Sie die komprimierte Eclipse-Setupdatei in ein Verzeichnis Ihrer Wahl, wie z.B. *C:* oder *C:\Compiler*. (Die extrahierten Dateien stehen dann automatisch in einem Unterverzeichnis *C:\eclipse* bzw. *C:\Compiler\eclipse*.)

   Eclipse gibt es ebenso wie das JDK sowohl als 32-Bit- wie auch als 64-Bit-Version. Achten Sie darauf, für Eclipse die gleiche Version zu installieren wie für das JDK!

   Achten Sie auch darauf, dass die in der ZIP-Datei enthaltenen Dateien unter Berücksichtigung der Pfadangaben extrahiert werden.

   Windows: Klicken Sie mit der rechten Maustaste auf die ZIP-Datei und wählen Sie im Kontextmenü den Befehl Alle extrahieren aus. Das Installationsverzeichnis, in das die Dateien extrahiert werden, können Sie anschließend in dem aufspringenden Dialogfeld eingeben.

   Wenn Sie ein spezielles ZIP-Programm installiert haben, kann es sein, dass der Befehl Alle extrahieren nicht mehr im Kontextmenü aufgeführt wird. Wahrscheinlich enthält das Menü dann aber Einträge, die das betreffende ZIP-Programm aufrufen.

   Extrahieren Sie Eclipse nicht in geschützte Verzeichnisse wie z.B. *C:\Programme*.

   Linux: Kopieren Sie die ZIP-Datei in das übergeordnete Verzeichnis, unter dem Eclipse installiert werden soll, und entpacken Sie die Datei, z.B.:

   ```
   tar -xzf eclipse-java-helios-SR2-linux-gtk.tar.gz
   ```

   Danach können Sie die ZIP-Datei löschen.

Fertig!

Deinstallation

Löschen Sie einfach das Verzeichnis der Eclipse-Installation.

1.4.2 Erster Start

Stufe 3: Eclipse (2. Phase)

1. Zum Starten von Eclipse rufen Sie einfach die Datei *eclipse.exe* aus dem Extraktionsverzeichnis (= Installationsverzeichnis) auf (z.B. durch Doppelklick im Windows Explorer).

Wenn Sie Eclipse das erste Mal starten, erscheint ein Dialogfeld, welches Sie nach dem zu verwendenden Workspace-Verzeichnis fragt. Die Vorgabe ist üblicherweise ein Verzeichnis namens *workspace*, das unter Ihrem Benutzerverzeichnis (unter Windows 7: *C:\Users\<Benutzername>*) angelegt wird.

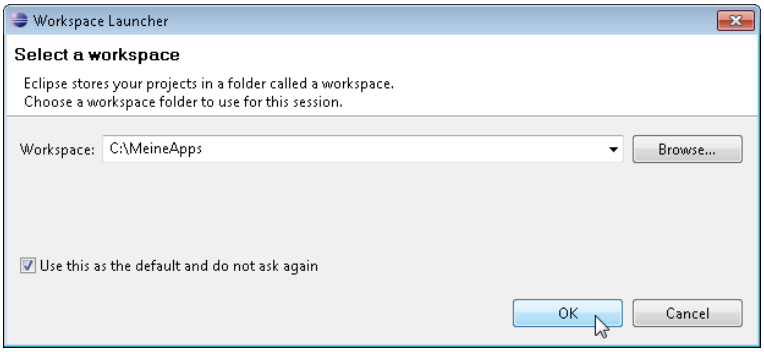


Abbildung 1.11: Das Arbeitsverzeichnis wird festgelegt.

2. Ersetzen Sie den vorgegebenen Pfad durch *C:\MeineApps* und aktivieren Sie die Option *Use this as the default ...*, um nicht bei jedem Start von Eclipse nach dem Workspace-Verzeichnis gefragt zu werden.
3. Klicken Sie auf OK.

Workspace (Arbeitsverzeichnis)

In Eclipse werden Programme (inklusive Apps), an denen Sie arbeiten, in Form von Projekten verwaltet. Dabei werden alle Dateien, die zu einem Projekt gehören (wie Quelltextdateien, Bilddateien etc.), in einem eigenen Verzeichnis abgelegt, dem sogenannten Projektverzeichnis, welches Sie angeben, wenn Sie ein neues Projekt anlegen.

Die Projekte, die Sie anlegen, werden selbst wieder in übergeordneten Arbeitsverzeichnissen, den sogenannten *workspaces*, abgelegt. Wenn Sie den Workspace wie oben beschrieben beim ersten Start von Eclipse festlegen, werden alle Projekte, die Sie später anlegen, unter diesem Workspace abgelegt.

Wenn Sie wieder beim Start von Eclipse nach dem zu verwendenden Workspace gefragt werden möchten, rufen Sie in Eclipse den Befehl Window/Preferences auf, wechseln Sie zur Seite General, Startup and Shutdown/Workspaces und aktivieren Sie die Option Prompt for workspace on startup.

Abbildung 1.12: Eclipse wurde das erste Mal gestartet.

Hinweis

Englischkenntnisse sind für die Arbeit mit der englischsprachigen Eclipse-IDE hilfreich. Eine Umstellung der IDE auf Deutsch ist unseres Wissens derzeit nicht möglich. (Vielleicht möchten Sie ja ein entsprechendes Plugin schreiben?) Für Leser, die mit der englischen Fachterminologie noch nicht so vertraut sind, haben wir daher im Anhang ein kleines Wörterbuch zusammengestellt.

4. Klicken Sie im Begrüßungsbildschirm auf das Symbol WORKBENCH.
5. Werfen Sie einen ersten Blick auf die Eclipse-IDE und schließen Sie danach das Programm.

Desktop-Verknüpfung anlegen

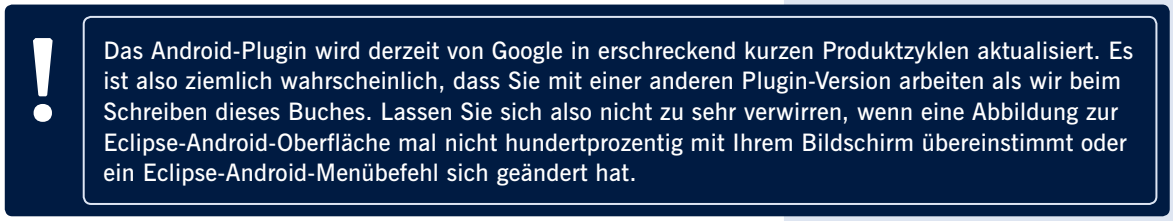
Da Sie in Zukunft wohl öfters mit Eclipse arbeiten werden, lohnt es sich, eine Desktop-Verknüpfung für das Programm anzulegen.

1. Starten Sie den Windows Explorer (⊞+E).
2. Wechseln Sie in das Eclipse-Verzeichnis.
3. Ziehen Sie die Datei *eclipse.exe* mit gedrückt gehaltener Maustaste auf Ihren Desktop-Hintergrund.
4. Drücken Sie die Tastenkombination Strg+⇧, sodass neben dem gezogenen Dateisymbol der Hinweis *Verknüpfung erstellen* erscheint, und lassen Sie dann die Maustaste los, um die Verknüpfung anzulegen.

1.5 Das Android-Plugin

Stufe 4: Plugin (1. Phase)

Stufe 4: Das Ende ist in Sicht. Jetzt müssen Sie nur noch unter Eclipse das Android-Plugin installieren.



Das Android-Plugin wird derzeit von Google in erschreckend kurzen Produktzyklen aktualisiert. Es ist also ziemlich wahrscheinlich, dass Sie mit einer anderen Plugin-Version arbeiten als wir beim Schreiben dieses Buches. Lassen Sie sich also nicht zu sehr verwirren, wenn eine Abbildung zur Eclipse-Android-Oberfläche mal nicht hundertprozentig mit Ihrem Bildschirm übereinstimmt oder ein Eclipse-Android-Menübefehl sich geändert hat.

1.5.1 Installation

1. Starten Sie Eclipse (siehe oben) und stellen Sie eine Internetverbindung her.
2. Rufen Sie den Menübefehl HELP/INSTALL NEW SOFTWARE auf.

   Es erscheint der Dialog AVAILABLE SOFTWARE.

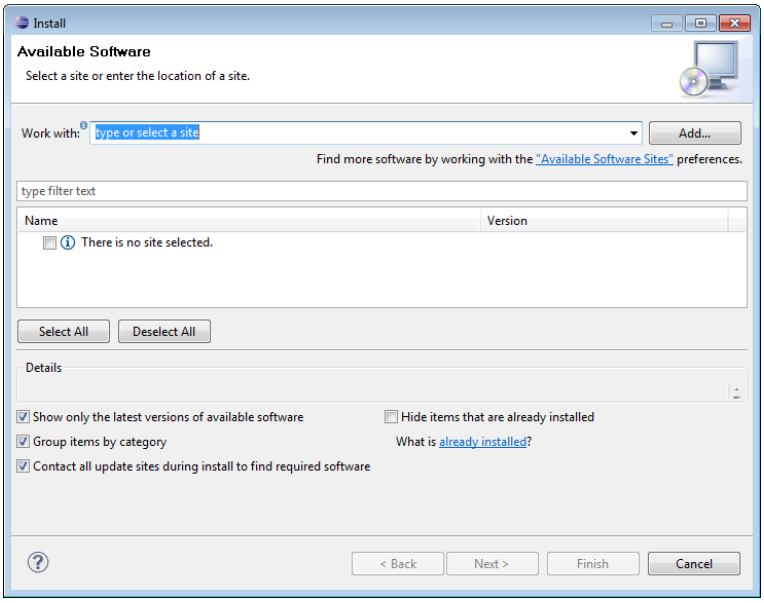


Abbildung 1.13: Über diesen Dialog werden Plugins integriert.

Das Dialogfeld ist anfangs leer und unbrauchbar. Um es zum Leben zu erwecken, müssen Sie eine Repository-Site einrichten.

3. Klicken Sie dazu rechts oben auf die Schaltfläche ADD.

   Es erscheint das Dialogfeld ADD REPOSITORY.

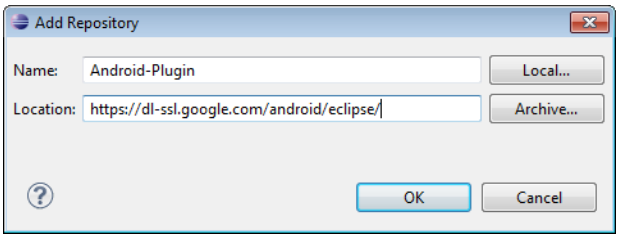


Abbildung 1.14: Geben Sie die Download-Adresse für das Android-Plugin ein.

4. Geben Sie in das Feld NAME einen Titel für die Repository-Site ein, z.B.: »Android-Plugin«.

   Der genaue Name spielt keine Rolle, sollte aber einen Hinweis darauf geben, welche Art von Software dahinter zu finden ist.

5. Geben Sie in das Feld LOCATION die nachfolgende Adresse ein und schicken Sie das Dialogfeld anschließend mit OK ab.

   https://dl-ssl.google.com/android/eclipse/

Nach Abschicken des Dialogs greift Eclipse auf die angegebene Site zu und zeigt das vorgefundene Angebot an. Im Dialogfeld sollte nun der Eintrag DEVELOPER TOOLS zu lesen sein.

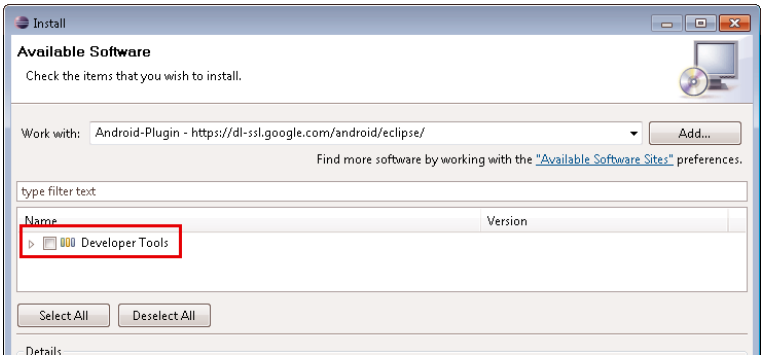


Abbildung 1.15: Die Verbindung zu Google-Repository-Site war erfolgreich.

6. Aktivieren Sie das Kontrollkästchen neben DEVELOPER TOOLS und klicken Sie auf die Schaltfläche NEXT.

Es erscheint das Dialogfeld INSTALL DETAILS, in dem die zu installierenden Komponenten noch einmal zusammengefasst sind. Achten Sie darauf, dass sich darunter ein Eintrag für die Android Development Tools, kurz ADT, findet.

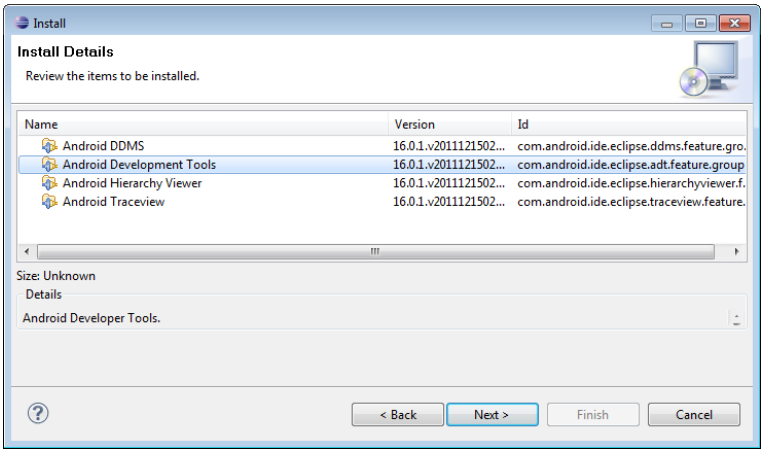


Abbildung 1.16: Die Liste der zu installierenden Komponenten

7. Klicken Sie auf NEXT, um die Komponenten installieren zu lassen.

Höchstwahrscheinlich werden Sie vorab Lizenzbedingungen und vermutlich auch das Installieren von nicht signierter Software bestätigen müssen.

8. Zum guten Schluss müssen Sie Eclipse neu starten.

1.5.2 Konfiguration

Stufe 4: Plugin (2. Phase)

Jetzt müssen Sie Eclipse nur noch mitteilen, wo es das Android-SDK findet.

1. Starten Sie, sofern nicht schon geschehen, Eclipse.
2. Rufen Sie den Menübefehl WINDOW/PREFERENCES auf.

   Es erscheint der Dialog PREFERENCES.
3. Wählen Sie links die Kategorie ANDROID aus.
4. Klicken Sie auf BROWSE und wählen Sie in dem erscheinenden Dialog das Installationsverzeichnis Ihres Android-SDK aus.
5. Klicken Sie dann auf APPLY ...

   Für jede der Android-Plattformen, die Sie mit dem SDK heruntergeladen und installiert haben, wird nun ein Target (Ziel für die App-Erstellung) eingerichtet.
6. ... und anschließend auf OK.

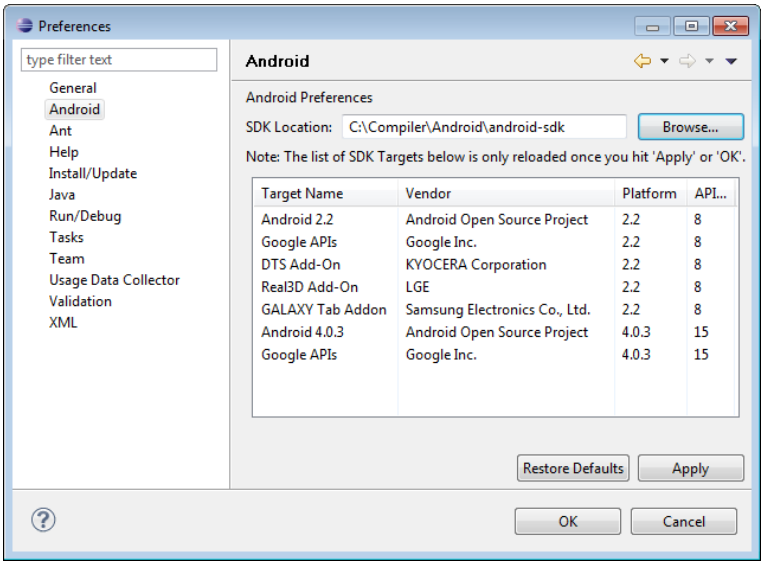

Preferences
type filter text
General
Android
Ant
Help
Install/Update
Java
Run/Debug
Tasks
Team
Usage Data Collector
Validation
XML
Android
Android Preferences
SDK Location: C:\Compiler\Android\android-sdk
Browse...
Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.
Target Name | Vendor | Platform | API...
Android 2.2 | Android Open Source Project | 2.2 | 8
Google APIs | Google Inc. | 2.2 | 8
DTS Add-On | KYOCERA Corporation | 2.2 | 8
Real3D Add-On | LGE | 2.2 | 8
GALAXY Tab Addon | Samsung Electronics Co., Ltd. | 2.2 | 8
Android 4.0.3 | Android Open Source Project | 4.0.3 | 15
Google APIs | Google Inc. | 4.0.3 | 15
Restore Defaults
Apply
OK
Cancel

Abbildung 1.17: Der letzte Klick

Glückwunsch! Ihr Rechner ist nun fertig eingerichtet. Und zur Belohnung werden Sie schon im *nächsten Kapitel* Ihre erste App erstellen.

1.6 Wo Sie weitere Hilfe finden

Weitere Informationen finden Sie auf der Support-Site zu diesem Buch, *www.carpelibrum.de*, und auf den Websites von Oracle (*www.oracle.com*), Eclipse (*www.eclipse.org*) und Android (*developer.android.com*).

Sollten die Hinweise im Buch und auf der Website nicht ausreichen, haben Sie keine Scheu, sich per E-Mail an uns zu wenden (*autoren@carpelibrum.de*).

1.7 Nächste Schritte

Der Rechner ist vorbereitet, die nötigen Hilfsmittel sind zusammengetragen, die Programmierumgebung ist eingerichtet. Nun wollen wir sehen, wie wir mithilfe dieser Programmierumgebung Apps erstellen, mit welchen Aufgaben wir bei der App-Programmierung konfrontiert werden und wie Apps grundsätzlich aufgebaut sind.

Für Programmieranfänger und Umsteiger von anderen Programmiersprachen ist jetzt ein guter Zeitpunkt, um mit dem Java-Schnellkurs auf der Buch-

CD zu beginnen. Programmieranfänger sollten die Kapitel 1 bis 3 des Java-Tutoriums auf der Buch-CD lesen. Umsteiger von anderen objektorientierten Sprachen können Kapitel 3 überspringen (sofern sie mit Begriffen wie Klassen, Instanzierung, Konstruktor, Methoden und Vererbung vertraut sind).

Wenn Sie möchten, können Sie auch bereits Kapitel 4 des Java-Tutoriums durcharbeiten, in dessen Zuge das Grundgerüst einer Java-Anwendung analysiert und das Konzept der Java-Bibliotheken und -Pakete vorgestellt wird. Es steht Ihnen aber auch frei, erst mit *Kapitel 2* dieses Buches zu beginnen und das Tutoriumskapitel 4 dann parallel zu *Kapitel 2.3*, »Das vorgegebene Codegerüst« zu lesen.

2 Auf die Plätze, fertig … App!

Sie lernen in diesem Kapitel,

- *wie Sie für eine App ein Eclipse-Projekt anlegen,*
- *wie das Grundgerüst einer App aussieht,*
- *wie Sie eine Ausgabe erzeugen,*
- *wie Sie Apps testen.*

Um eine eigene App zu schreiben, müssen Sie sich mit Ihrer Entwicklungsumgebung auskennen, Sie müssen die Grundlagen der Programmierung mit Java beherrschen und nicht zuletzt müssen Sie natürlich auch noch mit den speziellen Erfordernissen und Techniken der Android-Programmierung vertraut sein. Dies sind gewaltige Anforderungen.

So schwierig, wie es sich jetzt vielleicht anhört, ist es allerdings auch nicht. Wer einen Berg abtragen will, muss mit dem ersten Spatenstich beginnen. Blicken Sie also nicht auf den Berg, der noch vor Ihnen liegt, sondern immer nur auf den nächsten Spatenstich. Wir werden es mit unseren Erläuterungen ebenso halten.

2.1 Die Ruhe vor dem Sturm

Gerade der Einstieg in die App-Erstellung konfrontiert den Anfänger mit vielen neuen und ungewohnten Konzepten. Sie mit all diesen Konzepten direkt vertraut zu machen, wäre langwierig, ermüdend und nur wenig lehrreich.

Die gute Nachricht ist: Sie müssen sich nicht erst intensiv mit allen diesen Konzepten auseinandersetzen, um eine funktionierende App zu schreiben. Wir werden es daher im Folgenden so halten, dass wir uns jeweils nur auf die Konzepte, Techniken und Syntaxformen konzentrieren, die für den jeweils nächsten Schritt auf dem Weg zur angestrebten App wichtig sind. So wird niemand überfordert, es stellen sich schnell Erfolge ein, der Spaß an der App-Programmierung bleibt erhalten und wir dürfen dennoch sicher sein, dass sich nach und nach alles zu einem kompletten Gesamtbild zusammenfügt.

Holen Sie also noch einmal tief Luft … und los geht´s.

Die App, die wir in diesem Kapitel schreiben werden, soll nichts weiter tun, als uns auf dem Smartphone mit einem freundlichen »Hallo Programmierer« zu begrüßen. Das ist nicht gerade viel, aber es geht auch gar nicht darum, im ersten Versuch gleich eine sinnvolle und perfekte App zu erstellen. Es geht darum, einen Überblick über den App-Entwicklungsprozess zu bekommen. Und es geht darum, uns selbst zu beweisen, dass wir fähig sind, eigene Apps zu schreiben. Wenn wir diese App meistern, dann können wir auch jede andere App programmieren.

2.2 Das Projekt

Alles beginnt mit einem Projekt

Wer wie wir Eclipse als Entwicklungsumgebung verwendet, für den beginnt die Arbeit an jeder neuen App mit dem Anlegen eines passenden Projekts. Starten wir also Eclipse und legen wir ein neues Projekt an.

Projekte, Projekttypen und Wizards

Projekte sind die Eclipse-interne Verwaltungseinheit, in der Eclipse alle Daten und Dateien zusammenfasst, die nötig sind, um aus einem Quelltext eine App zu erstellen (oder auch ein Java-Konsolenprogramm, eine C++-Programmbibliothek oder was auch immer Sie programmieren möchten).

Unterschiedliche Programmierziele – möchten Sie eine App schreiben, ein normales Java-Programm oder »nur« eine Klassenbibliothek? – erfordern unterschiedlich konfigurierte Projekte. Um Ihnen die Konfiguration zu erleichtern, gibt es in Eclipse die sogenannten *Wizards*: dialogbasierte Assistenten, die darauf spezialisiert sind, Ihnen bei der Einrichtung von Projekten eines bestimmten Typs zu helfen. (Welche Wizards Ihnen im Einzelnen zur Verfügung stehen, hängt davon ab, welche Plugins Sie installiert haben.)

Auch für die App-Programmierung gibt es einen passenden Wizard. Er wurde zusammen mit dem Android-Plugin installiert und hilft uns dabei, ein App-Projekt anzulegen. Der Android-Wizard fügt unserem Projekt auch gleich diverse Quelldateien und Code hinzu, sodass wir als Ausgangspunkt eine einfache, aber ausführbare App erhalten.

1. Rufen Sie im Menü den Befehl File/New/Project auf.

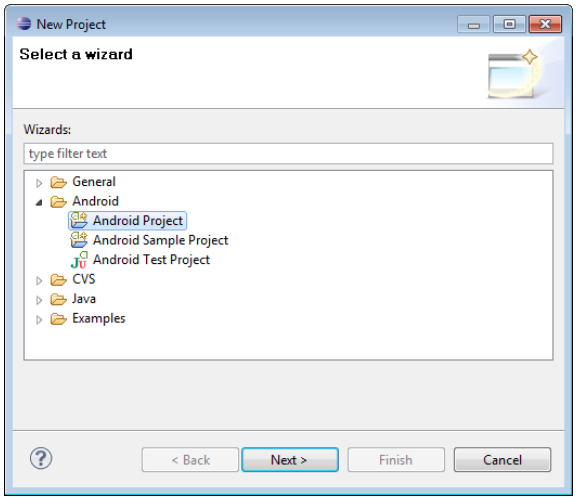


Abbildung 2.1: Ein neues Android-Projekt beginnen

2. Wählen Sie im Dialogfeld NEW PROJECT unter dem Ordner ANDROID den Eintrag ANDROID PROJECT und klicken Sie auf NEXT.

Es erscheint das Dialogfeld NEW ANDROID PROJECT.

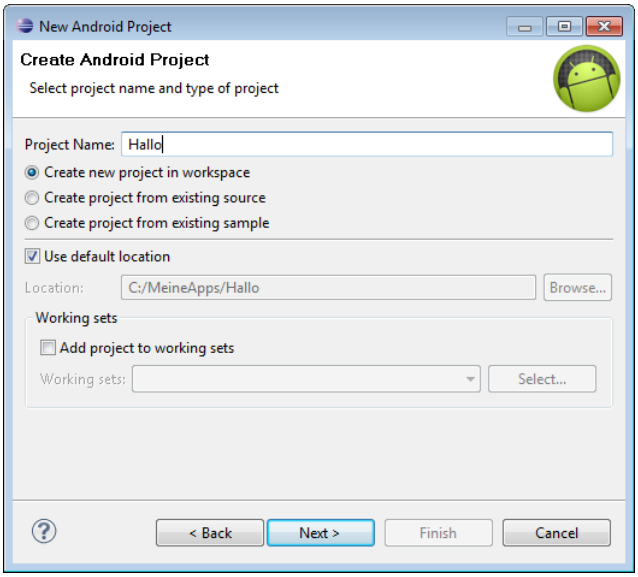


Abbildung 2.2:
Das Dialogfeld zum Anlegen neuer Projekte – einmal direkt nach dem Aufruf und einmal ausgefüllt für unser Hallo-Projekt

3. Geben Sie als Namen für das Projekt *Hallo* ein und klicken Sie auf NEXT.

   Unter diesem Namen wird Ihr Projekt fortan in Eclipse aufgeführt.

   Außerdem legt Eclipse auf der Festplatte, unter dem Verzeichnis des aktuellen Workspace ein Verzeichnis dieses Namens als Projektverzeichnis an. Jedenfalls dann, wenn Sie das Kontrollkästchen USE DEFAULT LOCATION aktiviert lassen.

   Verwenden Sie für den Projektnamen keine Sonderzeichen und möglichst auch keine Umlaute oder Leerzeichen.

   Achten Sie darauf, dass die Option CREATE NEW PROJECT IN WORKSPACE aktiviert ist, damit Eclipse ein neues Projektverzeichnis unter dem Verzeichnis des aktuellen Workspace anlegt.

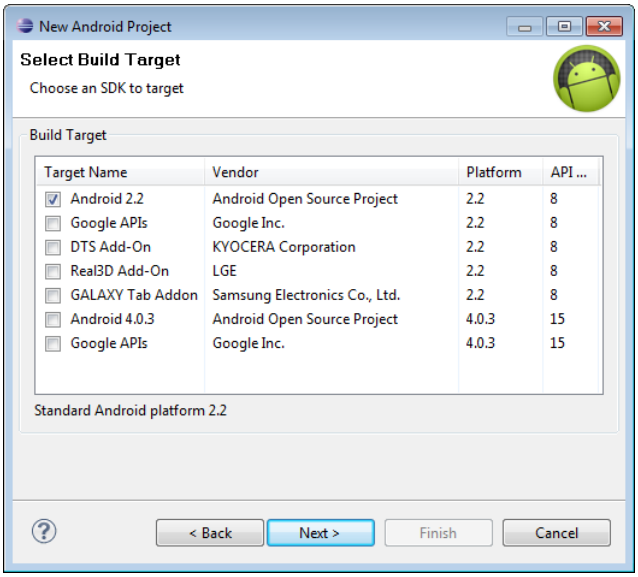
4. Auf der nächsten Seite des Dialogfelds markieren Sie die gewünschte Zielplattform. Für unsere Programme ist dies immer Android 2.2.

   Das Build Target gibt an, für welche Android-Version (Target) Eclipse Ihr Projekt später erstellen (Build) soll.

   Wir arbeiten in diesem Buch standardmäßig mit der Version 2.2, da diese einen guten Kompromiss aus Leistungsfähigkeit und Verbreitung darstellt, siehe auch Kasten.

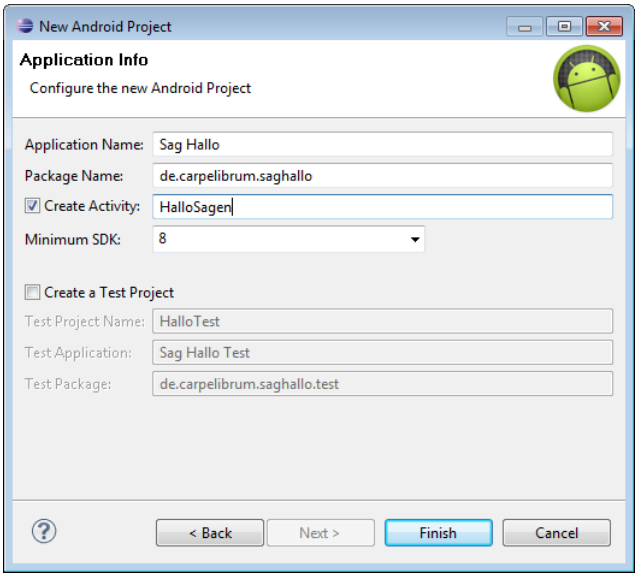
Die Anordnung der Befehle im Eclipse-Menüsystem kann je nach Eclipse-Version und installierten Plugins variieren. Falls Sie statt des obigen Befehls einen Menübefehl FILE/NEW/ANDROID PROJECT vorfinden, kann dies allerdings auch daran liegen, dass Ihr Workspace nicht korrekt eingerichtet ist (siehe Erläuterungen im Anhang zu Eclipse).

Abbildung 2.3:
Die Zielplattform auswählen



Welche Build Targets im Dialogfeld aufgelistet werden, hängt übrigens davon ab, welche Plattformen Sie mit dem SDK-Manager heruntergeladen haben (siehe *Kapitel 1.5.2*).

Abbildung 2.4:
Das dritte und letzte Seite des Dialogfelds New Android Project



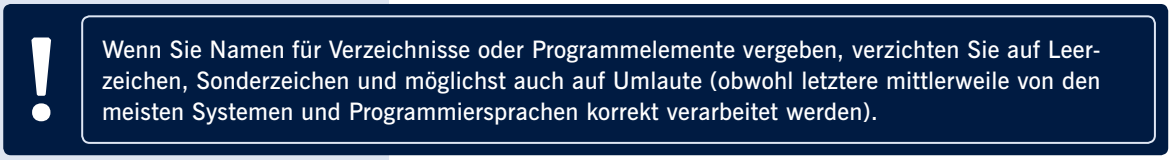
5. Nun muss noch das dritte und letzte Dialogfeld gefüllt werden. Da die Einstellungen etwas umfangreicher sind, haben wir sie in *Tabelle 2.1* zusammengefasst – inklusive der für unser Beispielprojekt erforderlichen Ein- und Ausgabe, sowie Erläuterungen, was sich hinter diesen Einstellungen verbirgt.

Tabelle 2.1: Einstellungen für das New Android Project-Dialogfeld

Feld	Eingabe	Bedeutung
Application Name	*Sag Hallo*	Unter diesem Namen wird Ihre App später auf dem Smartphone geführt.
Package Name	*ihrname.saghallo*	Pakete sind ein Element der Sprache Java, das zur Organisation des Codes genutzt wird. Die Angabe ist für App-Projekte zwingend erforderlich und muss eindeutig sein, um sicherzustellen, dass es auf einem Android-System keine zwei Apps mit gleichem Paketnamen gibt. Typisch ist daher die Verwendung von Domainnamen in umgekehrter Reihenfolge mit angehängtem App-Namen (also z.B. *de.meinewebsite.hallo*). Solange Sie aber nur Apps für den Eigenbedarf schreiben, sollte eine Kombination aus Ihrem Namen und dem Namen des Apps genügen. (Wir verwenden in unseren Beispielprojekten den Namen *de.carpelibrum* plus dem jeweiligen App-Namen.) Übrigens: Paketnamen werden traditionell klein geschrieben.
Create Activity	*HalloSagen*	Apps definieren üblicherweise eine oder mehrere Activities, die ausgeführt werden können. Wenn Sie das Kontrollkästchen Create Activity aktivieren, legt Eclipse für die neue App gleich eine Activity (Aktivität) des vorgegebenen Namens an. Was diese Activity macht, müssen Sie aber natürlich später selbst programmieren. Achtung! Der Name darf keine Leer- und keine Sonderzeichen enthalten, da das Android-Plugin unter diesem Namen eine Klasse definiert. (In Java dürfen Namen von im Code definierten Elementen keine Leer- oder Sonderzeichen enthalten.)

Tabelle 2.1: Einstellungen für das New Android Project-Dialogfeld (Forts.)

Feld	Eingabe	Bedeutung
Min SDK Version	*8*	Die Mindestversion des SDK muss zu dem ausgewählten Build Target passen. Wenn Sie das Feld freilassen, wird automatisch die dem Target entsprechende SDK-Version ausgewählt (also 8 für Android 2.2). Wenn Sie die Mindest-SDK-Version explizit angeben, darf diese allerdings nicht größer sein als die API-Version, die dem ausgewählten Target entspricht (siehe Tabelle in Dialogfeld).


Wenn Sie Namen für Verzeichnisse oder Programmelemente vergeben, verzichten Sie auf Leerzeichen, Sonderzeichen und möglichst auch auf Umlaute (obwohl letztere mittlerweile von den meisten Systemen und Programmiersprachen korrekt verarbeitet werden).

6. Klicken Sie auf Finish, um das Projekt anlegen zu lassen.

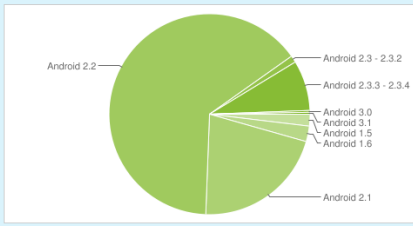
   Wenn Sie zuerst noch einmal auf Next klicken, erhalten Sie die Möglichkeit, zu dem neuen Projekt auch gleich noch ein passendes Testprojekt anzulegen. In diesem Buch werden wir von dieser Möglichkeit allerdings keinen Gebrauch machen.

Plattformen und APIs

Wie von jeder Software kommen auch von dem Android-Betriebssystem ständig neue und erweiterte Versionen (Plattformen) heraus. Zu jeder dieser Plattformen gibt es eine eigene API[1]. Der SDK-Manager lädt per Voreinstellung die Bibliotheken zu allen Android-Versionen herunter, sodass Sie später, wenn Sie beginnen, eine App zu schreiben, frei auswählen können, zu welcher Android-Version die App kompatibel sein soll.

Doch welche Version soll es sein? Grundsätzlich gilt: Je höher die Version, umso mehr technische Features werden von der Programm-API unterstützt. Eine höhere Version bedeutet aber auch, dass es draußen mehr Smartphones gibt, die diese Version nicht unterstützen. (Die Plattformen sind abwärtskompatibel, d.h. ein Smartphone mit Android 3.0 kann eine 2.2-App ausführen, aber ein Smartphone mit Android 2.2 kann umgekehrt keine 3.0-App ausführen.) Wichtig ist ferner die Unterscheidung, ob die App für ein Smartphone gedacht ist (Versionen Android 2.x) oder speziell für ein Tablet (Android 3.x). Google plant, das Versionsgewirr etwas zu verbessern, und wird mit Android 4.0 eine einheitliche API für alle Arten von Android-Geräten bereitstellen.

Ein guter Kompromiss aus Leistungsfähigkeit und Verbreitung sind derzeit die Plattformen 2.2 und 2.3. Wir haben uns im Hinblick auf eine möglichst breite Verfügbarkeit für die Version 2.2 entschieden, die im Juni 2011 auf ungefähr 75 % der Smartphones lauffähig war.



Platform	API Level	Distribution
Android 1.5	3	1.9%
Android 1.6	4	2.5%
Android 2.1	7	21.2%
Android 2.2	8	64.6%
Android 2.3 - Android 2.3.2	9	1.1%
Android 2.3.3 - Android 2.3.4	10	8.1%
Android 3.0	11	0.3%
Android 3.1	12	0.3%

Data collected during a 14-day period ending on June 1, 2011

Abbildung 2.5:
Aktuelle Verteilung der Android-Plattformen (Screenshot der Webseite http://developer.android.com/resources/dashboard/platform-versions.html)

1 API ist das Akronym für »Application Programming Interface« und die Schnittstelle zu den Bibliotheksklassen, die Sie für die App-Programmierung benutzen.

Automatische Fehlerkontrolle

Hoppla, da stimmt etwas nicht

Gerade für Einsteiger sehr hilfreich ist die automatische Fehlerkontrolle des *New Android Project*-Dialogs, die eine ganze Reihe von fehlerbehafteten oder widersprüchlichen Einstellungen abfängt.

Beispielsweise ist der Paketname ein Java-Bezeichner und darf daher – anders als z.B. der Anwendungsname (*Application name*) keine Leerzeichen enthalten. Falls Sie dennoch Leerzeichen einbauen, beispielsweise in der Form »carpe librum.hallo«, weist Sie das Dialogfeld mit einer Meldung im Kopfbereich darauf hin. Zusätzlich wird die Schaltfläche FINISH deaktiviert, sodass Sie gezwungen sind, den Fehler zu beheben, bevor Sie das Projekt anlegen lassen können.

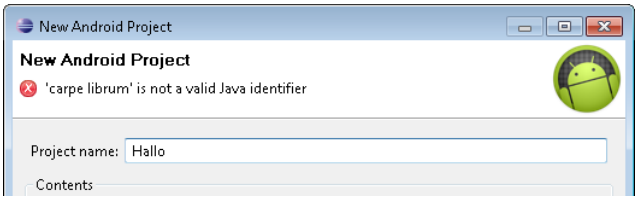


Abbildung 2.6: *Das Dialogfeld weist Sie darauf hin, dass es in den vorgenommenen Einstellungen eine Unstimmigkeit gibt.*

Ein weiterer Fehler, der vom Dialogfeld automatisch erkannt wird, ist eine Diskrepanz im angegebenen Build Target und der Min SDK-Version.

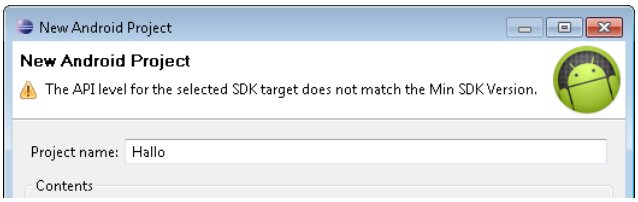

New Android Project
New Android Project
The API level for the selected SDK target does not match the Min SDK Version.
Project name:
Hallo
Contents

Abbildung 2.7: *Diese Meldung erscheint, wenn die angegebene Min SDK-Version kleiner ist als die API-Level des ausgewählten Build Target.*

Falls Sie den Package Explorer aus Versehen geschlossen haben oder er aus irgendeinem anderen Grund nicht angezeigt wird, können Sie ihn jederzeit durch Aufruf des Menübefehls WINDOW/SHOW VIEW/PACKAGE EXPLORER wieder anzeigen lassen.

Das Projekt in Eclipse

Zurück im Hauptbildschirm von Eclipse sehen Sie das neu angelegte Projekt. Anfänglich nur als einzelner unscheinbarer Eintrag im *Package Explorer*, doch wenn Sie auf den Namen doppelklicken, wird der Projektknoten expandiert und es zeigt sich, dass das Projekt bereits aus einer umfangreichen Sammlung von Ordnern und Dateien besteht.

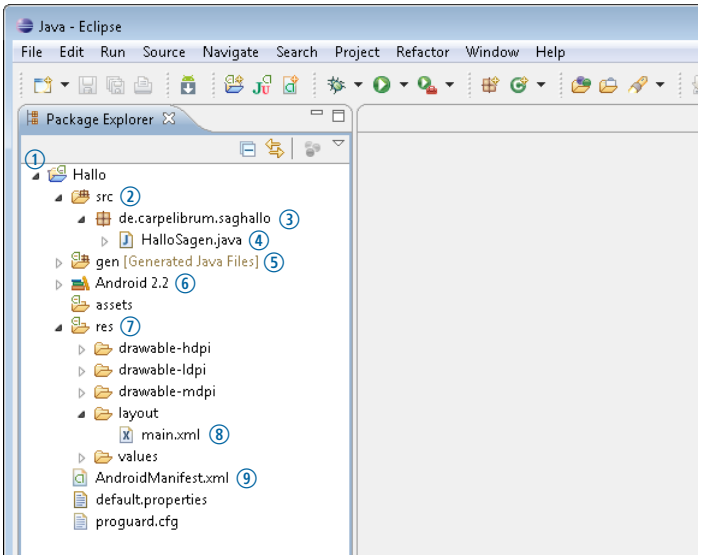


Abbildung 2.8:
Das neu angelegte Projekt

① *Der Projektknoten*
② *Ordner für die Quelltextdateien*
③ *Für das Paket gibt es einen eigenen Unterknoten*
④ *Die Java-Quelltextdatei – sie enthält den eigentlichen Code*
⑤ *Ordner für automatisch generierte Dateien, die Sie in der Regel nicht direkt bearbeiten*
⑥ *Die Bibliotheken für das anvisierte Android-Target*
⑦ *Ordner für Ressourcen, unter anderem ...*
⑧ *... in XML definierte Layouts*
⑨ *Die Manifestdatei listet die einzelnen Bestandteile der App*

Das Projekt auf der Festplatte

Die Ordnerstruktur des Projekts im Package Explorer finden Sie fast identisch auch auf der Festplatte wieder. Aus dem vorangehenden Abschnitt wissen Sie ja bereits, dass Eclipse zu jedem Projekt ein gleichnamiges Projektverzeichnis auf der Festplatte anlegt. Aber auch die Unterordner der Projektstruktur finden Sie als Unterordner im Projektverzeichnis wieder.

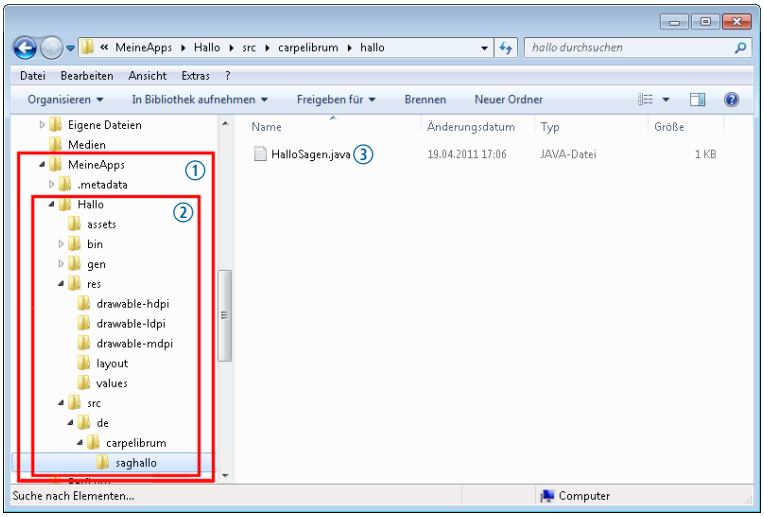


Abbildung 2.9: Das Projekt Hallo auf der Festplatte

① *Der Workspace MeineApps*
② *Das Projekt Hallo*
③ *Die Quelltextdatei*

Eine kleine Abweichung gibt es lediglich bei dem Paketnamen. Java verlangt nämlich, dass für jeden der durch Punkte getrennten Namensteile eines Paketnamens ein eigenes Unterverzeichnis erstellt wird und dass die Quelldateien aus einem Paket in dem durch den Paketnamen spezifizierten Verzeichnis stehen. Aus diesem Grund finden Sie die Quelltextdatei *HalloSagen.java* aus unserem Paket `de.carpelibrum.saghallo` nicht direkt im Unterverzeichnis *Hallo/src*, sondern unter *Hallo/src/de/carpelibrum/saghallo*.

Path:	/Hallo
Type:	Project
Location:	C:\MeineApps\Hallo

In den Properties kann das Projektverzeichnis nachgeschlagen werden

Sie erinnern sich nicht mehr, in welchem Workspace-Verzeichnis Ihr Projekt liegt? Dann klicken Sie im Package Explorer mit der rechten Maustaste auf den Projektknoten und wählen Sie den Befehl Properties. Wählen Sie links im Properties-Dialogfeld den Eintrag Resource und Sie können rechts unter Location nachlesen, wo das Projektverzeichnis liegt.

Auch wenn der Code des für uns angelegten Codegerüsts nur wenige Zeilen umfasst, nutzt er bereits eine Vielzahl objektorientierter und Java-typischer Syntaxelemente. Programmieranfänger und Umsteiger von anderen Sprachen sollten daher jetzt Kapitel 4 des Java-Tutoriums von der Buch-CD öffnen und das Tutorium vorab oder parallel zu den folgenden Ausführungen lesen.

2.3 Das vorgegebene Codegerüst

Lassen Sie uns nun einen Blick in den Code werfen, der bereits für uns erzeugt wurde.

1. Doppelklicken Sie im Package Explorer auf den Knoten der Quelltextdatei *HalloSagen.java*.

Eclipse lädt den Inhalt der Datei in den Editor.

Listing 2.1: Inhalt der Datei HalloSagen.java

```
package de.carpelibrum.saghallo;

import android.app.Activity;
import android.os.Bundle;

public class HalloSagen extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Leser, die bereits über Erfahrung in der Java-Programmierung verfügen, werden die grundlegende Struktur sicher schnell entschlüsselt haben ... und sich vielleicht wundern, dass keine `main()`-Methode definiert wird.

Auf Leser, die noch über keinerlei Erfahrung in der java-Programmierung verfügen, dürfte dieses Codebeispiel eher abschreckend wirken, doch der Aufbau ist gar nicht so kompliziert, wie es auf den ersten Blick aussieht. Blendet man die Details aus, schälen sich drei bis vier Komponenten heraus (siehe Abbildung 2.8).

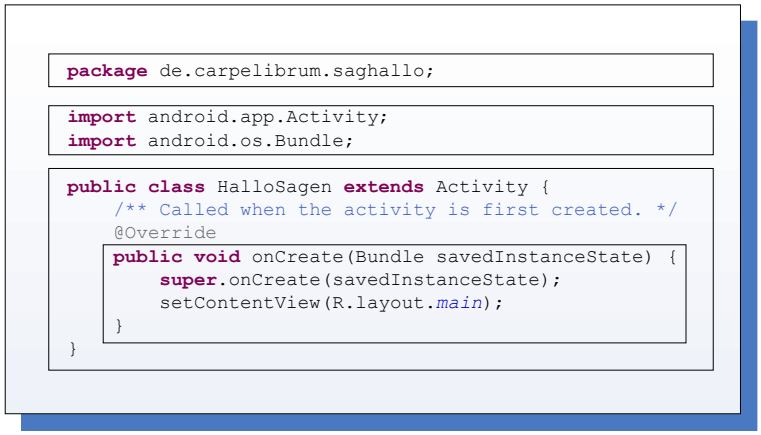


Abbildung 2.10: Struktur des Code-Grundgerüsts

2.3.1 Die package-Anweisung

In der obersten Zeile treffen wir den Paketnamen wieder, den wir im Dialogfeld New Android Project eingegeben haben.

Die Anweisung

```
package de.carpelibrum.saghallo;
```

sorgt dafür, dass alle nachfolgend definierten Codeelemente – in unserem Grundgerüst wäre dies die Klasse `HalloSagen` – untergeordnete Elemente des Pakets `de.carpelibrum.saghallo` sind.

Pakete

Die Android-Programmierung verlangt, dass alle Klassen einer App in einem gemeinsamen Paket zusammengefasst werden. Außerdem sollte der verwendete Paketname, der beim Anlegen des Android-Projekts angegeben wird, eindeutig sein, damit es auf den Smartphones nicht zu Namenskonflikten durch Klassen aus gleichlautenden Paketen kommt.

Die gute Nachricht ist, dass Eclipse-Anwender mit der Paketverwaltung kaum aktiv zu tun haben. Sie geben den Paketnamen einfach bei der Einrichtung des Projekts an (siehe *Abschnitt 2.2*), und um alles Weitere kümmert sich Eclipse.

Mehr zur Codeorganisation mit Paketen erfahren Sie in Kapitel 4 des Java-Tutoriums auf der Buch-CD.

2.3.2 Die import-Anweisungen

Unter der Paketangabe stehen zwei `import`-Anweisungen.

```
import android.app.Activity;
import android.os.Bundle;
```

Die Namen `android.app` und `android.os` sind Pakete aus der Android-Bibliothek. In dieser Bibliothek gibt es zahlreiche Klassen, die wir für die Programmierung unserer Apps verwenden können.

Zwei dieser Klassen sind z.B. `Activity` und `Bundle`. Die Klasse `Activity` ist dabei im Paket `android.app` und die Klasse `Bundle` im Paket `android.os` definiert.

Klassennamen beginnen in Java üblicherweise mit einem Großbuchstaben (während Paketnamen immer klein geschrieben werden).

import-Anweisungen

Die Aufgabe von `import`-Anweisungen ist es, Klassennamen in unseren Code zu importieren. Dies vereinfacht den Code und spart uns Tipparbeit.

Grundsätzlich ist es nämlich so, dass Klassen, die in Paketen definiert sind, nur über ihren vollständigen Namen (der die Paketangabe einschließt) angesprochen werden können – also z.B. mit:

```
public class HalloSagen extends android.app.Activity
```

Wenn Sie einen Klassennamen importieren, können Sie auf die Paketangabe verzichten:

```
public class HalloSagen extends Activity
```

Mehr zur `import`-Anweisung erfahren Sie im Java-Tutorium, Kapitel 4.

2.3.3 Die Klassendefinition

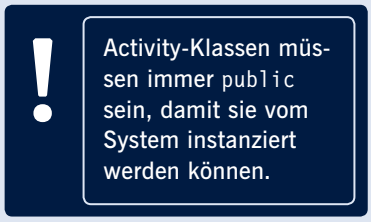
So langsam lassen wir den notwendigen Verwaltungskram hinter uns und kommen zu dem eigentlichen funktionellen Code, der in unserem Grundgerüst in der Klasse `HalloSagen` zusammengefasst ist.

Bevor wir uns den Code der Klasse allerdings etwas genauer ansehen, müssen wir den Begriff der »Activity« einführen. Als Anwender sind Sie gewohnt, dass Sie mit Apps (jedenfalls mit den meisten) interagieren können – beispielsweise, indem Sie die Schaltfläche einer App drücken oder in einem Spiel ein Raumschiff mit dem Finger nach links oder rechts bewegen. Alle diese Interaktionen laufen über die visuelle Benutzeroberfläche der App (das, was Sie auf dem Touchscreen Ihres Android-Geräts sehen). Apps organisieren diese Benutzeroberfläche in einzelne Bildschirmseiten. Jede dieser Bildschirmseiten erfüllt eine bestimmte Aufgabe oder »Aktivität« (*Activity*) – eine zusammengehörende Gruppe von Interaktionen, wie z.B. das Ausfüllen eines E-Mail-Formulars oder das Anschauen einer Diashow.

Activity = Bildschirmseite + zugehörigem Code

Erinnern Sie sich noch an unsere Einstellungen im Dialogfeld New Android Project? Dort haben wir Eclipse mitgeteilt, dass es für unsere App schon einmal eine erste Activity namens `HalloSagen` einrichten soll. Diese Activity finden wir nun im Code wieder.

Im Code sind Activities Klassen, die von der Basisklasse `Activity` abgeleitet sind und unter anderem über eine Methode `onCreate()` verfügen, die automatisch aufgerufen wird, wenn die Activity vom Anwender oder dem System das erste Mal gestartet wird.



```
public class HalloSagen extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Wenn der Anwender die App startet, instanziert das System die Klasse `HalloSagen` und führt die Methode `onCreate()` aus.

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
```

Ausführliche Informationen zur Programmierung mit Klassen und Methoden finden Sie in den Kapiteln 5 bis 9 des Java-Tutoriums.

Die erste Zeile ruft die Methode `onCreate()` der Basisklasse (`Activity`) auf. Auf diese Weise werden alle in der Basisklasse definierten Initialisierungsarbeiten automatisch für uns ausgeführt.

Die zweite Anweisung ruft die ebenfalls von `Activity` geerbte Methode `setContentView()` auf. Diese Methode legt das Layout unserer App fest, sprich den Aufbau ihrer Benutzeroberfläche.

@Override

Sicher ist Ihnen das Symbol `@Override` aufgefallen. Es handelt sich dabei um eine Java-Annotation – eine an den Compiler gerichtete Anmerkung –, die in diesem Fall darauf aufmerksam machen soll, dass die nachfolgend definierte Methode eine gleichnamige Methode aus der Basisklasse überschreiben soll. (Zur Erläuterung des Prinzips der Überschreibung siehe Kapitel 9 des Java-Tutoriums.)

Überschreibung funktioniert aber nur, wenn die Methode mit dem gleichen Namen und der gleichen Signatur definiert wird. Gibt es Abweichungen, wird stattdessen eine neue Methode definiert. Im Falle der `onCreate()`-Methode wäre dies fatal, denn es würde bedeuten, dass der von uns vorgesehene Code nicht beim Start der Activity ausgeführt wird.

Um derartigen Unbill zu vermeiden, setzt man die Annotation `@Override`. Der Compiler prüft dann, ob tatsächlich eine Überschreibung vorliegt. Falls nicht, gibt er eine Fehlermeldung aus.

2.4 Layout und Ressourcen

Jede App verfügt über ein Layout, das bestimmt, wie die Benutzeroberfläche aussieht. Mit anderen Worten: Das Layout legt im Wesentlichen fest, was der Anwender auf dem Touchscreen sieht, wenn er die App startet und ausführt.

Aus Sicht des Programmierers bedeutet Layout, dass er im Code die Elemente erzeugt, aus denen die Oberfläche zusammengesetzt ist. Die meisten Java-Programmierer sind es gewohnt, hierfür Code zu schreiben. Das heißt, sie suchen sich die Klassen zusammen, die die gewünschten Elemente definieren, erzeugen von diesen Klassen Objekte und konfigurieren diese. Obwohl diese Vorgehensweise auch in der Android-Programmierung gangbar ist ...

```
package de.carpelibrum.saghallo;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;    // Klasse für Textfelder

public class HalloSagen extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        TextView tv = new TextView(this);
        tv.setText("Hallo Programmierer");
        setContentView(tv);
    }
}
```

... ist es in Android üblich, das Layout über XML-Code zu definieren – eine Technik, die mehr und mehr auch in der traditionellen Anwendungsprogrammierung Anwendung findet.

Statisches Layout = XML

2.4.1 XML-Layouts

Das Programmgerüst, das für unsere App angelegt wurde, verfügt bereits über ein erstes, einfaches XML-Layout, das nach Belieben angepasst oder ausgetauscht werden kann. Wie dies geht, lesen Sie in *Kapitel 5*. Für unsere erste App werden wir das vorgegebene Layout einfach beibehalten. Wir werden uns allerdings schon einmal ansehen, wie die Technik des XML-Layouts grundsätzlich funktioniert und wie wir ein solches Layout konfigurieren können.

Werfen wir noch einmal einen Blick in den Code unserer App:

```
package de.carpelibrum.saghallo;

import android.app.Activity;
import android.os.Bundle;

public class HalloSagen extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Die letzte Zeile der Methode `onCreate()`, die die `Activity`-Methode `setContentView()` aufruft, richtet die Benutzeroberfläche der App ein. Sie können dieser Methode als Argument wahlweise ein Objekt einer View-Klasse übergeben oder eine ID, die eine Layoutdatei angibt. Letzteres geschieht im obigen Code. Die ID lautet in diesem Fall `R.layout.main`.

Nur einsehen, nicht ändern: R.java

`R` ist der Name einer Klasse, die automatisch für das Projekt erzeugt wird und deren Aufgabe die **R**essourcenverwaltung ist (daher der Name `R`). Diese Klasse enthält als Elemente eine Reihe von untergeordneten Klassen, darunter eben auch die Klasse `layout`, in der schließlich die ID `main` definiert ist.

Die ID führt den Compiler zu einer Datei *main.xml*, die im Projektunterverzeichnis *res/layout* liegt.

Wenn Sie wissen möchten, welcher konkrete Wert sich hinter der ID `main` *verbirgt, erstellen Sie das Projekt, expandieren Sie im Package Explorer die Knoten gen und de.carpelibrum.saghallo und doppelklicken Sie auf den Knoten für die Datei R.java, um diese in den Editor zu laden.*

1. Expandieren Sie im Package Explorer die Knotenfolge *res/layout* und doppelklicken Sie auf den Eintrag für die Datei *main.xml*, um diese in den Editor zu laden.

   Für XML-Layoutdateien gibt es zwei unterschiedliche Ansichten, die über Reiter am unteren Rand des Editors ausgewählt werden. Per Voreinstellung zeigt Eclipse immer erst die Ansicht GRAPHICAL LAYOUT.

2. Klicken Sie am unteren Rand des Editors auf den Reiter MAIN.XML, um die Layoutdatei in der XML-Ansicht zu betrachten.

Listing 2.2: Inhalt der Layoutdatei main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

Wir haben es hier mit einem hierarchischen Layout zu tun. Auf der obersten Ebene steht eine LinearLayout-Komponente. In diese Komponente ist eine TextView-Komponente eingebettet.

Layout-Komponenten haben vor allem die Aufgabe, die in ihnen eingebetteten Komponenten nach einer bestimmten Strategie auszurichten. Die Strategie der LinearLayout-Komponente ist, die eingebetteten Komponenten nebeneinander auszurichten (und gegebenenfalls umzubrechen).

Die TextView-Komponente erlaubt die Anzeige eines Textes.

Für jedes dieser XML-Elemente wird bei der Kompilierung die entsprechende Android-Klasse instanziert und das so erzeugte Objekt gemäß den in den XML-Elementen definierten Attributen konfiguriert.

Für das TextView-Element wurde z.B. festgelegt, wie breit es ist (Attribut `android:layout_width`), das sein Inhalt bei Bedarf in die nächste Zeile umbrochen werden soll (Attribut `android:layout_height`) und welchen Text es anzeigen soll (Attribut: `android:text`).

Welche Werte Sie den Attributen zuweisen können, hängt von der Art des Attributs ab (mehr dazu in *Kapitel 5*). Uns interessiert im Moment vor allem das `text`-Attribut des TextView-Elements. Ihm wird nicht der auszugebende Text übergeben, sondern ein Verweis auf eine Ressource, in der der Text gespeichert ist.

2.4.2 Ressourcen

In der App-Programmierung ist es üblich, Texte und Bildelemente, die auf der Benutzeroberfläche angezeigt werden, als Ressourcen zu verwalten.

Für Textelemente gibt es dazu z.B. die Datei *strings.xml*.

Strings

Texte, die ein Java-Programm ausgibt oder anderweitig verarbeitet, werden in der Programmierung als »Strings« bezeichnet.

3. Expandieren Sie im Package Explorer die Knotenfolge *res/values* und doppelklicken Sie auf den Eintrag für die Datei *strings.xml*, um diese in den Editor zu laden.

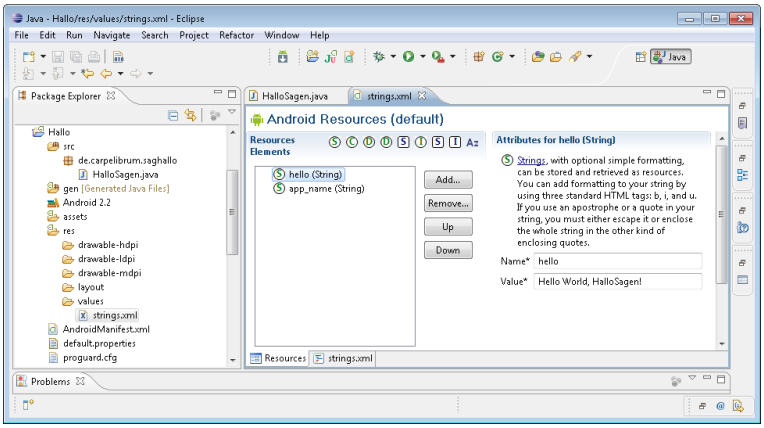
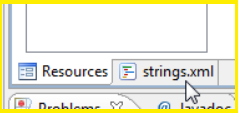


Abbildung 2.11: Die Datei strings.xml im Ressourceneditor



Die Reiter für Ressourcen- und Textansicht

In Eclipse können Sie die Ressourcendatei *strings.xml* wahlweise über die Eingabemaske des Ressourceneditors (siehe Abbildung 2.9) oder als reinen XML-Code bearbeiten.

Um zwischen Ressourceneditor und Texteditor hin und her zu schalten, klicken Sie einfach am unteren Rand des Editorfenster auf die zugehörigen Reiter.

Listing 2.3: Inhalt der Datei strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HalloSagen!</string>
    <string name="app_name">Sag Hallo</string>
</resources>
```

Um den auszugebenden Text von »Hello World, HalloSagen!« in »Hallo Programmierer!« zu ändern, geben Sie den neuen Text einfach zwischen den Tags des ersten `string`-Elements ein ...

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hallo Programmierer!</string>
    <string name="app_name">Sag Hallo</string>
</resources>
```

... oder klicken Sie im Ressourceneditor auf den Eintrag *hello* und tippen Sie dann rechts den neuen Text in das Eingabefeld VALUE ein.

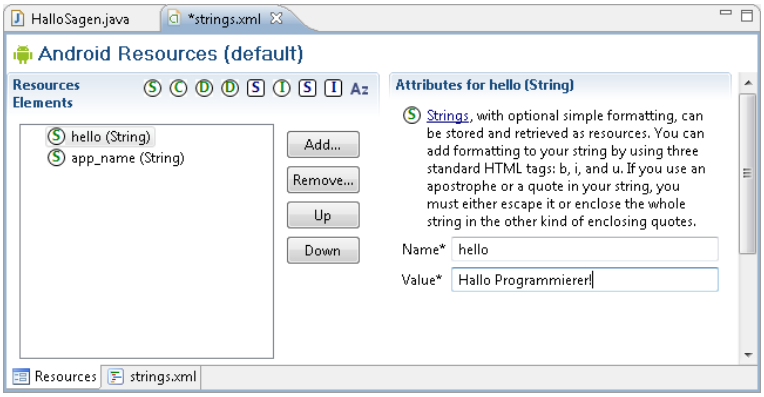

HalloSagen.java
*strings.xml
Android Resources (default)
Resources
Elements
hello (String)
app_name (String)
Add...
Remove...
Up
Down
Attributes for hello (String)
Strings, with optional simple formatting, can be stored and retrieved as resources. You can add formatting to your string by using three standard HTML tags: b, i, and u. If you use an apostrophe or a quote in your string, you must either escape it or enclose the whole string in the other kind of enclosing quotes.
Name*
hello
Value*
Hallo Programmierer!
Resources
strings.xml

Abbildung 2.12: Bearbeitung einer String-Ressource im Ressourceneditor

4. Speichern Sie Ihre Änderungen, indem Sie den Menübefehl File/Save All aufrufen.

2.5 Die App erstellen (Build)

Eclipse ist standardmäßig so eingestellt, dass Änderungen an einem Projekt sofort kompiliert und ausgewertet werden. Dies ist bequem, kann aber auch manchmal nerven, vor allem, wenn ungewiss bleibt, ob es mit der Erstellung geklappt hat oder nicht. Lesern, die mit Eclipse noch nicht so vertraut sind, empfehlen wir daher, die Option vorerst zu deaktivieren und ihre Projekte manuell zu erstellen.

Wir ziehen die manuelle App-Erstellung vor

1. Öffnen Sie das Project-Menü.
2. Wenn vor dem Befehl Build Automatically ein Häkchen zu sehen ist, wählen Sie den Befehl aus, um die Option zu deaktivieren und das Häkchen verschwinden zu lassen.

Um Ihr Projekt von Hand zu erstellen, gehen Sie wie folgt vor:

1. Speichern Sie das Projekt mit dem Menübefehl File/Save All (oder durch Klick auf die entsprechende Symbolschaltfläche).

   Dieser Schritt ist ganz wichtig, da sonst die letzten Änderungen beim Kompilieren nicht berücksichtigt werden.

2. Rufen Sie den Menübefehl Project/Build Project auf.

   Alternativ können Sie im Package Explorer von Eclipse mit der rechten Maustaste auf den Projektknoten Hallo klicken und im Kontextmenü den Befehl Build Project aufrufen.

Gibt es Fehler im Projekt, markiert Eclipse die betroffenen Stellen im Editor (rote Wellenlinie und Lampensymbol mit rotem Kästchen in Randleiste). Ob Code betroffen ist, der gerade nicht im Editor angezeigt wird, können Sie im

rot = Fehler
gelb = Warnung

Oft bietet das QuickInfo-Fenster unter dem Fehlertext einen oder mehrere Optionen an, wie der Fehler behoben werden könnte. Diese Korrekturvorschläge führen allerdings nicht immer zum gewünschten Ziel! Klicken Sie nur dann auf eine der angebotenen Optionen, wenn Sie sicher sind, dass der Fehler damit in Ihrem Sinne behoben wird.

Package Explorer kontrollieren. Enthält eine Datei Fehler, werden die Datei und alle übergeordneten Ordner mit einem roten Kästchen mit weißem Kreuz markiert.

Fehler in Quelltexten sollten Sie immer von oben nach unten abarbeiten, da es gut möglich ist, dass es sich bei den unteren Fehlern um Folgefehler handelt, die nach Beheben der oberen Fehler und erneuter Erstellung automatisch verschwinden.

Sehen Sie sich die fehlerhaften Stellen an. Können Sie den Fehler erkennen? Wenn nicht, fahren Sie mit der Maus über die als fehlerhaft markierte Quelltextzeile, um sich ein QuickInfo-Fenster mit näheren Erläuterungen zu dem Fehler anzeigen zu lassen. Beheben Sie den Fehler, speichern Sie und lassen Sie das Projekt neu erstellen.

Tipp

Fehler, die bei der Erstellung auftreten, werden auch im Fenster Problems *(Aufruf über Menübefehl* Window/Show View/Problems*) aufgelistet.*

2.6 Die App im Emulator testen

Zeit, dass wir uns einmal ansehen, wie unsere App aussieht, wenn sie ausgeführt wird. Wir müssen sie dazu nicht erst auf ein Android-Smartphone übertragen, ja wir müssen nicht einmal Eclipse verlassen. Ein Befehl genügt, und Eclipse lädt den Emulator und überträgt auf diesen die App. Im Fenster des Emulators können wir die App dann starten und testen.

Doch damit dies alles so reibungslos und einfach funktioniert ... müssen wir dem Emulator zuerst mitteilen, welchen Typus von Android-Gerät er simulieren soll.

2.6.1 AVD für Emulator anlegen

AVD = virtuelles Android-Gerät

Um Apps auf Ihrem Rechner ausführen zu können, müssen Sie ein virtuelles Android-Gerät anlegen – quasi also ein virtuelles Smartphone –, das der Emulator simulieren soll. Doch keine Angst, dies ist nicht die Fortsetzung des Installationsmarathons aus dem *vorangehenden Kapitel*. Alles, was zur Einrichtung eines solchen virtuellen Geräts – Android spricht von einem »Android Virtual Device« (kurz AVD) – nötig ist, haben Sie bereits in *Kapitel 1.3* mit dem SDK-Manager heruntergeladen.

1. Starten Sie den Android-SDK-Manager.

   Sie können den AVD-Manager direkt von Eclipse aus starten, indem Sie auf sein Symbol in der Symbolleiste klicken.

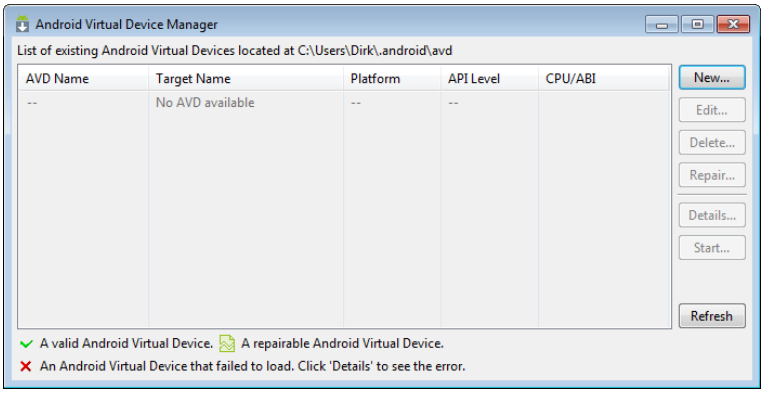


Abbildung 2.13:
Anlegen eines Emulators (Virtual Device) im SDK-Manager

2. Klicken Sie auf die Schaltfläche NEW.

   Es erscheint das Dialogfeld CREATE NEW ANDROID VIRTUAL DEVICE.

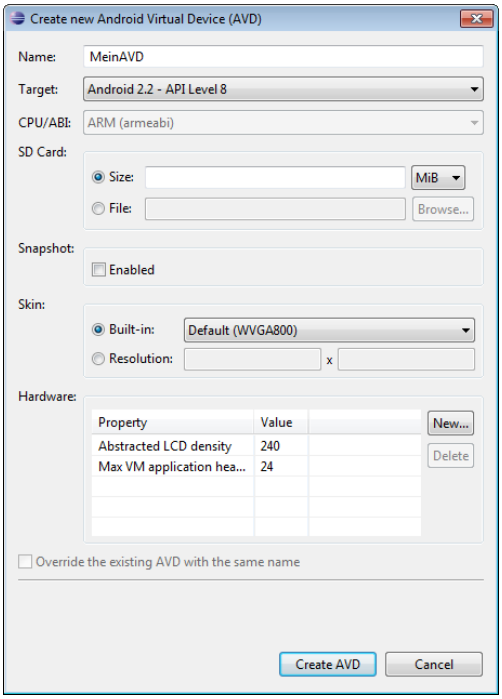


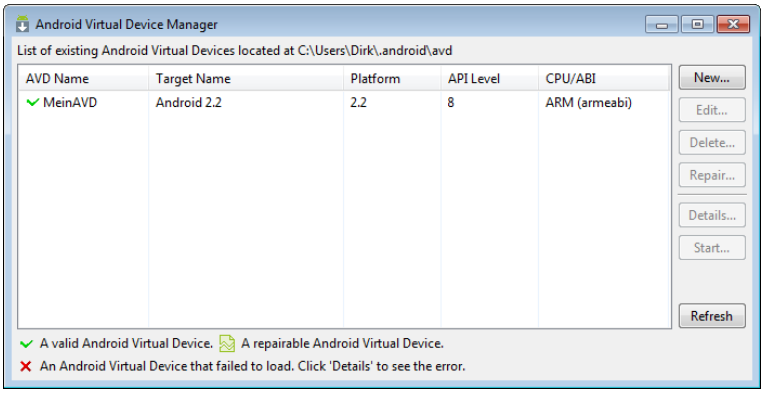
Abbildung 2.14:
Konfiguration des neu anzulegenden AVD

3. Geben Sie einen Namen für das AVD ein, z.B. *MeinAVD*.
4. Wählen Sie als Target die Plattform *Android 2.2* aus.

   Das Listenfeld führt alle Plattformen auf, für die Sie zuvor mit dem SDK-Manager die SDK heruntergeladen und installiert haben (siehe *Kapitel 1.3*).

5. Behalten Sie unter Skin die Einstellung Built-in bei.
6. Drücken Sie auf Create AVD.
7. Schließen Sie den SDK-Manager.

Abbildung 2.15:
Das AVD ist eingerichtet.



Aufsteiger

Zum Einarbeiten in die App-Programmierung genügt in der Regel ein einziges AVD. Sobald Sie aber beginnen, Apps an Bekannte weiterzugeben oder gar auf dem Android Market Place anzubieten, sollten Sie mehrere AVDs für unterschiedliche Plattformen (Einstellung Target), Auflösungen (Einstellung Skin) und Hardware-Ausstattungen (Einstellungen SD Card und Hardware) einrichten, um bequem testen zu können, wie sich Ihre App auf unterschiedlichen Android-Geräten verhält. (Tipp: Bauen Sie dann Informationen über Plattform, Auflösung und evtl. auch Ausstattung in den AVD-Namen ein.)

2.6.2 App testen

Jetzt ist es bald soweit, dass Sie Ihre App im Emulator begutachten können. Doch zuvor sollten Sie noch das Console-Fenster aufrufen, damit Sie den Startprozess des Emulators und der App live mitverfolgen können.

1. Rufen Sie den Befehl Window/Show View/Console auf, um das Console-Fenster einzublenden.
2. Rufen Sie den Befehl Run/Run auf, um die App zu testen.

   Alternativ können Sie auch im Package Explorer von Eclipse mit der rechten Maustaste auf den Projektknoten Hallo klicken und im Kontextmenü den Befehl Run As/Android Application aufrufen.

! Wenn Sie ein Projekt das erste Mal mit dem Befehl Run/Run ausführen, erscheint das Dialogfeld Run As, in dem Sie Android Application als Startmodus auswählen müssen. Eclipse erstellt daraufhin eine passende Launch-Konfiguration und benutzt diese fortan automatisch.

Im Console-Ausgabefenster von Eclipse erscheinen daraufhin nacheinander diverse Meldungen und auf Ihrem Touchscreen erscheint das Fenster des Emulators.

3. Warten Sie, bis im Console-Fenster die Meldung »ActivityManager: Starting: Intent ...« erscheint.

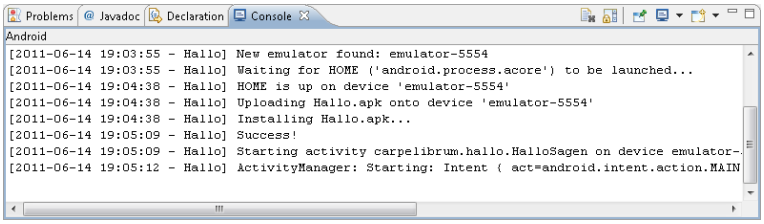

Problems
Javadoc
Declaration
Console
Android
[2011-06-14 19:03:55 - Hallo] New emulator found: emulator-5554
[2011-06-14 19:03:55 - Hallo] Waiting for HOME ('android.process.acore') to be launched...
[2011-06-14 19:04:38 - Hallo] HOME is up on device 'emulator-5554'
[2011-06-14 19:04:38 - Hallo] Uploading Hallo.apk onto device 'emulator-5554'
[2011-06-14 19:04:38 - Hallo] Installing Hallo.apk...
[2011-06-14 19:05:09 - Hallo] Success!
[2011-06-14 19:05:09 - Hallo] Starting activity carpelibrum.hallo.HalloSagen on device emulator-
[2011-06-14 19:05:12 - Hallo] ActivityManager: Starting: Intent { act=android.intent.action.MAIN

Abbildung 2.16:
Die App ist im Emulator und gestartet.

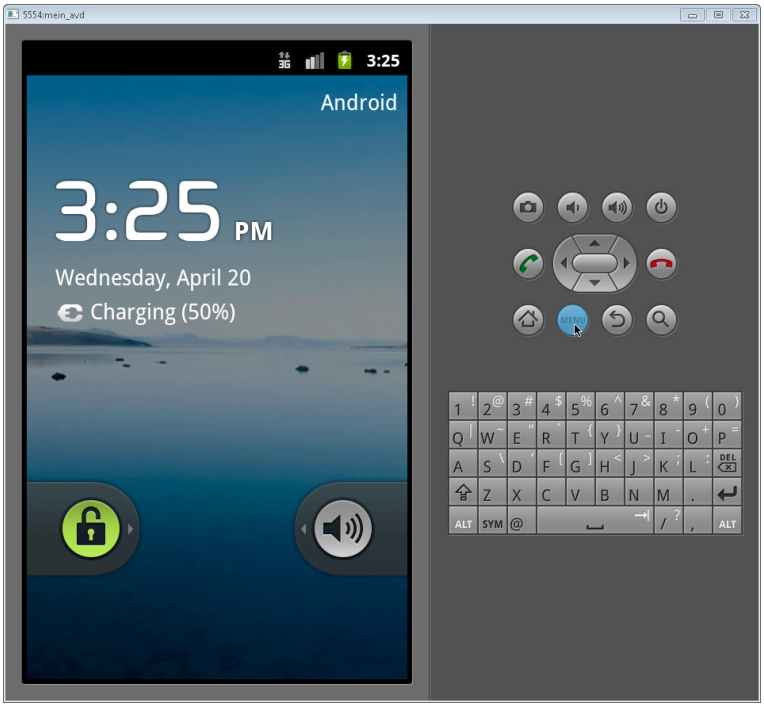


Abbildung 2.17:
Der Emulator ist einsatzbereit.

4. Klicken Sie falls nötig im Emulator auf die Schaltfläche Menu.

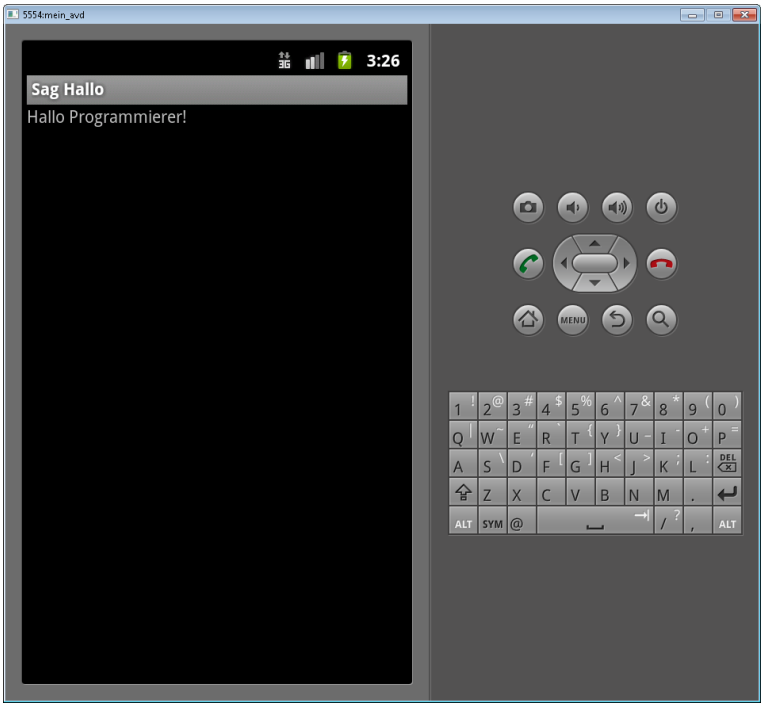


Abbildung 2.18: Die Benutzeroberfläche unserer App – mit Titelleiste und darunter gelegenem Textfeld

Hinweis

Ausführliche Informationen zum Emulator finden Sie in Anhang C.

Wie Sie sehen, wurde die `onCreate()`-Methode bereits ausgeführt und im Textfeld ist der von uns angepasste Ausgabetext zu lesen.

5. Beenden Sie die App, in dem Sie im Emulator auf die ZURÜCK-Taste klicken.
6. Schließen Sie das Emulatorfenster.

Tipp

Solange sie an einer App arbeiten, sollten Sie den einmal gestarteten Emulator nicht unnötig neu starten. Nicht, dass dies zu Problemen führen würde, es kostet nur sehr viel Zeit. Lassen Sie den Emulator also während der Arbeit an der App geöffnet und starten Sie das App einfach zum Testen mit dem Befehl Run/Run neu.

2.7 Die App auf dem Smartphone testen

Sie besitzen ein Android-Smartphone? Dann möchten Sie doch bestimmt auch gleich einmal sehen, wie sich Ihre App auf dem Smartphone macht? Nun, sofern Ihr Smartphone mit Android 2.2 oder höher läuft (der Android-Plattform, die wir beim Anlegen des Projekts als anvisierte Zielplattform (Target) eingestellt haben), sollte dies kein Problem sein.

Um die App auf einem realen Smartphone zu testen, gibt es zwei Möglichkeiten:

- den schnellen Weg mit Unterstützung durch Eclipse (setzt natürlich voraus, dass Eclipse auf dem Rechner installiert ist und Sie das Projekt der App vorliegen haben)
- den manuellen Weg, der immer funktioniert, vorausgesetzt, Sie haben eine APK-Datei vorliegen.

2.7.1 Automatische Übertragung mit Eclipse

Wenn Sie die App öfters zum Testen auf Ihr Smartphone übertragen möchten, können Sie Eclipse so einrichten, dass beim Ausführen der App von Eclipse aus das gewünschte Zielgerät (Emulator oder angeschlossenes Smartphone) abgefragt wird.

1. Stellen Sie Ihr Smartphone auf USB-Debugging um.

   Im Galaxy-Smartphone muss dazu unter EINSTELLUNGEN/ANWENDUNGEN/ENTWICKLUNG die Option USB-DEBUGGING aktiviert sein.

   Außerdem müssen Sie die Kies-Treiber von der Begleit-CD zu Ihrem Smartphone auf Ihrem Rechner installiert haben!

2. Stecken Sie das USB-Kabel ein.

3. Rufen Sie in Eclipse den Menübefehl RUN/RUN CONFIGURATIONS auf.

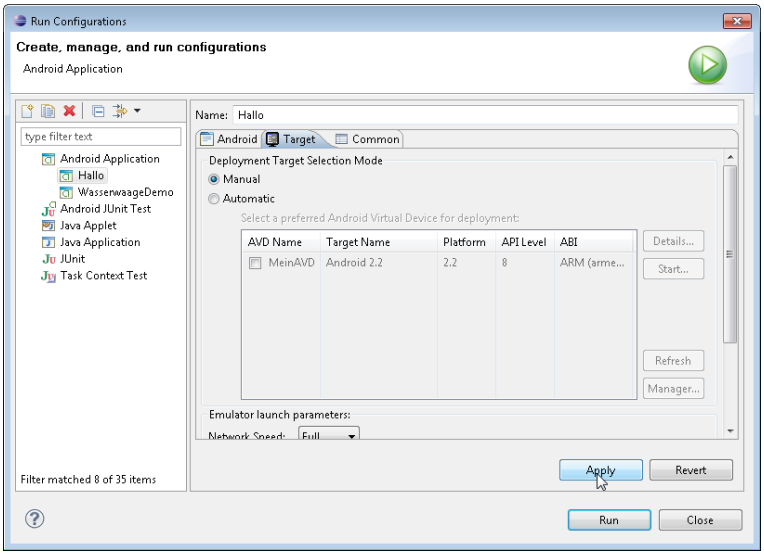


Abbildung 2.19: Eclipse-Konfiguration für die manuelle Auswahl des Zielgeräts

4. Wählen Sie, falls nötig, im linken Teil des Dialogfelds RUN CONFIGURATIONS unter der Kategorie ANDROID APPLICATION ihre App (in unserem Fall *Hallo*) aus.

5. Wechseln Sie im rechten Teil zur Registerseite TARGET und wählen Sie als DEPLOYMENT TARGET SELECTION MODE die Option MANUAL aus.

6. Klicken Sie auf APPLY und anschließend auf RUN.

Danach erscheint das Dialogfeld ANDROID DEVICE CHOOSER, in dem Ihr angeschlossenes Smartphone unte seiner Seriennummer aufgeführt sein sollte.

7. Wählen Sie das Smartphone mit einem Mausklick aus und drücken Sie dann OK.

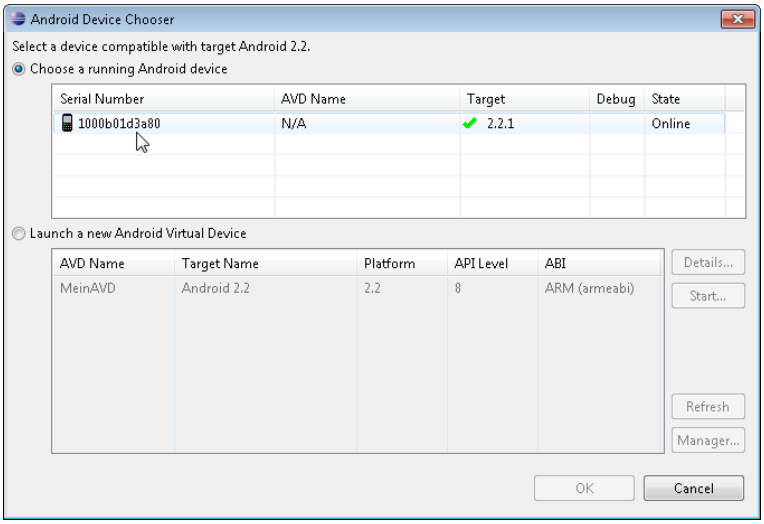


Abbildung 2.20:
Auswahl des angeschlossenen Smartphones als Zielgerät

Das Dialogfeld Android Device Chooser *wird fortan bei jeder Ausführung der App aufgerufen – solange, bis Sie die Launch-Konfiguration wieder auf* Automatic *umstellen.*

2.7.2 Manuelle Übertragung

Wenn Sie eine App von einem Rechner aus übertragen möchten, auf dem Eclipse nicht installiert ist oder kein App-Projekt verfügbar ist, können die App auch manuell übertragen.

Voraussetzung ist allerdings, dass Sie eine signierte APK-Datei der App zur Verfügung haben. Diese APK-Datei erstellen Sie am einfachsten mit Eclipse. Das Ganze artet damit in einem modernen Dreisprung aus: Auf der Laufbahn (Ihrem Entwicklungsrechner mit Eclipse) legen Sie die Ziel-SDK-Version fest und exportieren das Projekt in APK-Datei. Der letzte Sprung vom Trittbrett in die Sandgrube ist dann die Übertragung per USB, die von jedem beliebigen Rechner aus erfolgen kann.

Die Ziel-SDK-Version

Target-SDK ≥ Min-SDK

Um eine App exportieren zu können, müssen Sie mindestens zwei SDK-Versionen angeben:

- die Target-SDK-Version – dies ist die Version, für die Sie Ihre Apps eigentlich schreiben
- die Min-SDK-Version – dies ist die kleinste Version, auf der Ihre App noch lauffähig ist

Die Min-SDK-Version wurde bereits für uns gesetzt, nach Maßgabe unserer Angaben bei der Projekterzeugung.

Um die Ziel-SDK-Version festzulegen, müssen Sie die Manifestdatei des Projekts – dies ist die Datei *AndroidManifest.xml* – in den Editor laden und die Einstellung Uses Sdk bearbeiten.

1. Doppelklicken Sie im Package Explorer auf den Knoten der *AndroidManifest.xml*-Datei.
2. Klicken Sie in der Manifest-Ansicht der Datei im Bereich MANIFEST EXTRAS auf den Eintrag USES SDK.
3. Achten Sie darauf, dass die MIN SDK VERSION gesetzt ist und setzen Sie zudem auch die TARGET SDK VERSION.

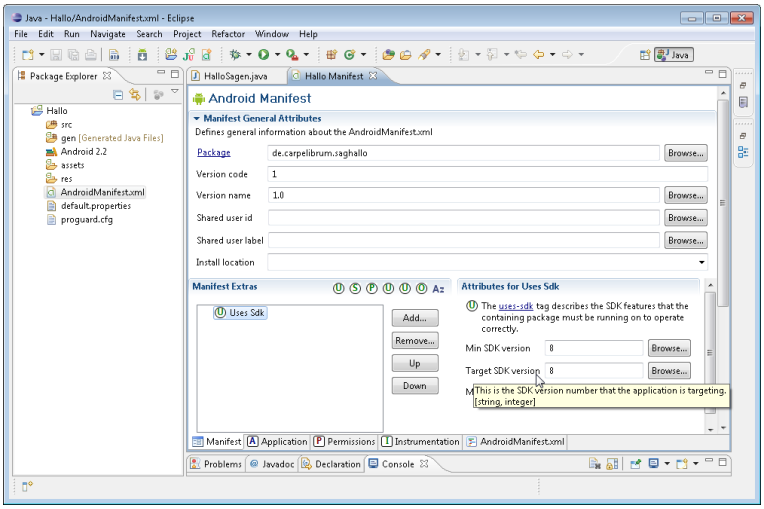


Abbildung 2.21: Festlegen der Target-SDK-Version

Das Projekt in eine APK-Datei exportieren

1. Rufen Sie den Menübefehl FILE/EXPORT auf.

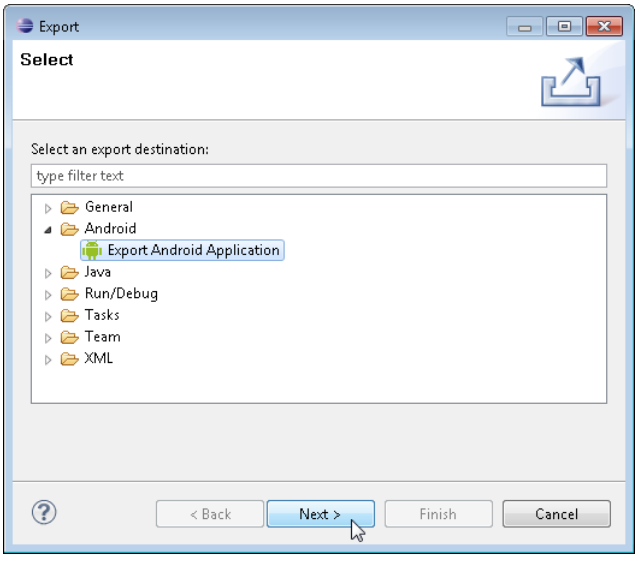


Abbildung 2.22: Export-Typ wählen

2. Wählen Sie im Dialog Export als Exporttyp die Option Android/Export Android Application aus und klicken Sie auf Next.
3. Wählen Sie im Dialog Export Android Application auf der Seite Project Checks das zu exportierende Projekt aus und klicken Sie auf Next.

Als Nächstes muss die App signiert werden. Zum lokalen Testen auf unserem eigenen Smartphone genügt uns dazu ein auf die Schnelle konstruierter Schlüssel.

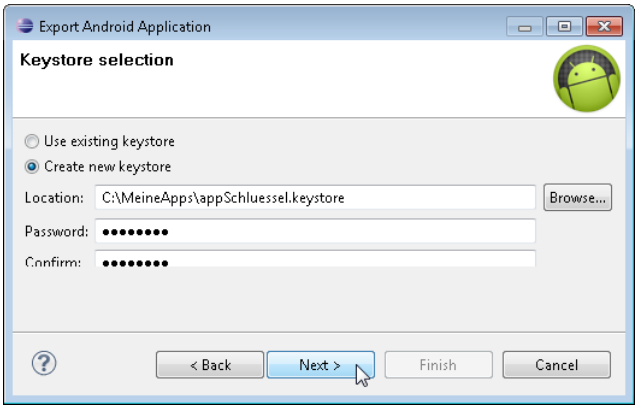


Abbildung 2.23: Anlegen eines Schlüssel-Speicherorts

Das erste Passwort ist für die keystore-Datei

4. Erzeugen Sie auf der Seite Keystore selection einen Speicherplatz für Ihren Schlüssel.

   Aktivieren Sie die Option Create new keystore. (Wenn Sie später weitere Schlüssel erstellen, können Sie diesen Speicherplatz wiederverwenden, Option Use existing keystore.)

   Geben Sie als Location einen Dateinamen samt Pfad und mit der Extension *keystore* an. Wenn die Datei noch nicht existiert, wird sie automatisch für Sie angelegt.

   Tippen Sie ein Passwort mit mindestens sechs Stellen für den Zugriff auf die Datei ein und wiederholen Sie es im Feld Confirm.

   Klicken Sie auf Next.

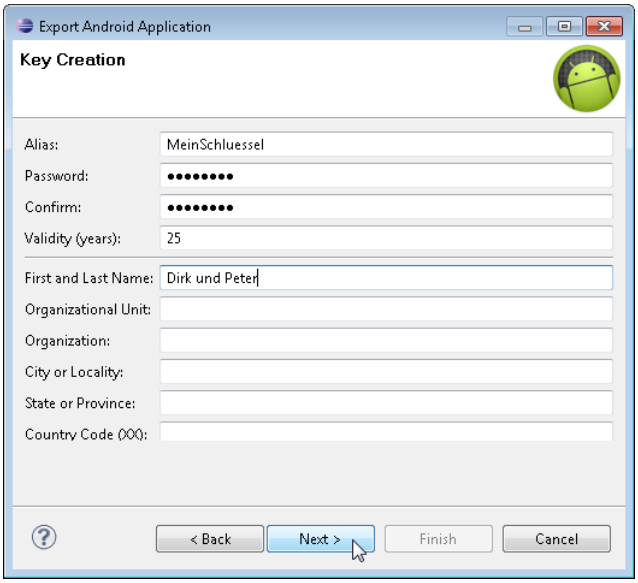


Abbildung 2.24:
Anlegen eines Schlüssels

5. Erzeugen Sie auf der Seite KEY CREATION einen Schlüssel.

   Geben Sie einen Namen (ALIAS) für den Schlüssel an.

   Tippen Sie ein Passwort mit mindestens sechs Stellen für den Zugriff auf den Schlüssel ein und wiederholen Sie es im Feld CONFIRM.

   Legen Sie fest, wie lange die Signierung gültig sein soll. Der empfohlene Wert ist 25.

   Füllen Sie mindestens eines der nach unten folgenden Felder aus und klicken Sie dann auf NEXT.

Das zweite Passwort ist für den Schlüssel

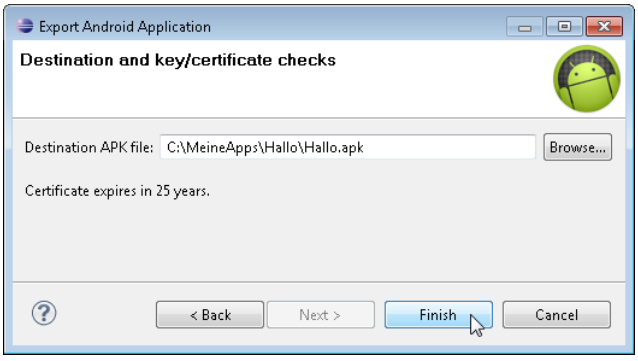


Abbildung 2.25:
Die APK-Datei wird erzeugt.

6. Legen Sie fest, wo die durch den Export erzeugte APK-Datei gespeichert werden soll, und klicken Sie auf Finish.
7. Kontrollieren Sie im Windows Explorer, ob die APK-Datei erzeugt wurde.

USB-Übertragung

Zum Schluss müssen sie die APK-Datei nur noch über einen USB-Anschluss in das Dateisystem Ihres Smartphones übertragen. Wie Sie dazu im Detail vorgehen müssen, erfahren Sie (hoffentlich[1]) im Benutzerhandbuch zu Ihrem Smartphone.

Für das Samsung Galaxy S (GT-I9000) gehen Sie z.B. wie folgt vor:

1. Prüfen Sie vorab, ob folgende Einstellungen gesetzt sind:

   Stellen Sie sicher, dass unter Einstellungen/Drahtlos und Netzwerk/USB-Einstellungen die Option Bei Verbindung fragen ausgewählt ist.

   Stellen Sie sicher, dass unter Einstellungen/Anwendungen die Option Unbekannte Quellen ausgewählt ist.

   Stellen Sie sicher, dass unter Einstellungen/Anwendungen/Entwicklung die Option USB-Debugging deaktiviert ist.
2. Verbinden Sie das Smartphone mit einem USB-Anschluss Ihres Rechners.

   Es erscheint die Bildschirmseite USB-Modus auswählen.
3. Wählen Sie die Option Massenspeicher aus.
4. Drücken Sie auf der Seite USB-Massenspeicher auf USB-Speicher verbinden.

Jetzt können Sie vom Rechner aus auf das interne Dateisystem des Smartphones zugreifen.

5. Legen Sie unter dem Verzeichnis *Android/data* ein Verzeichnis für Ihre App an, z.B. *de.IhrName*.
6. Kopieren Sie die APK-Datei in das Verzeichnis.
7. Drücken Sie auf der Seite USB-Massenspeicher des Smartphones auf Ausschalten.
8. Trennen Sie Smartphone und Rechner.
9. Klicken Sie im App-Menü auf *Eigene Dateien* und wechseln Sie in das App-Verzeichnis mit der APK-Datei.
10. Klicken Sie auf die APK-Datei der App und lassen Sie die App installieren.

1 Manche Smartphone-Benutzerhandbücher sind mit Hilfestellungen bei eher technischen Fragen oder fortgeschrittenen Anwendungsmöglichkeiten sehr sparsam.

2.8 Nächste Schritte

Sie haben nun bereits Ihre erste App erstellt und ausgeführt. Dabei sind eine ganze Menge von neuen Konzepten auf Sie eingeströmt und sicher von Ihrer Seite auch etliche Fragen aufgetaucht. Unsere nächste Aufgabe wird daher sein, die drängendsten Fragen zu klären und etwas mehr Licht auf die angesprochenen Konzepte zu werfen.

Leser, für die dies der Einstieg in die Java-Programmierung überhaupt ist, sollten sich vor dem Weiterlesen die Zeit nehmen, sich etwas intensiver mit Java vertraut zu machen – beispielsweise indem Sie den Praxisteil des Java-Schnellkurses auf der Buch-CD durcharbeiten.